Leibniz-Rechenzentrum

der Bayerischen Akademie der Wissenschaften

Technischer Bericht

Description of Goods and Services for the next Supercomputer in Bavaria (HLRB II)

Reinhold Bader, Matthias Brehm, Ralf Ebner, Helmut Heller, Herbert Huber, Hans-Ulrich Schäfer, Horst-Dieter Steinhöfer, Frank Wagner

Okt 2004 **LRZ-Bericht 2004-04**

Direktorium:
Prof. Dr. H.-G. Hegering (Vorsitzender)
Prof. Dr. A. Bode
Prof. Dr. Chr. Zenger

Leibniz-Rechenzentrum
Barer Straße 21
D-80333 München
UST-ID-Nr. DE811305931

Telefon: (089) 289-28784
Telefax: (089) 2809460
E-Mail: lrzpost@lrz.de
Internet: http://www.lrz.de

Öffentl. Verkehrsmittel:
U2, U8: Königsplatz
U3, U4, U5, U6: Odeonsplatz
Tram 27: Karolinenplatz

1 General information on the RFP

1.1 General objectives for the HLRB II

The supercomputer to be acquired, "Höchstleistungsrechner in Bayern II" (HLRB II), will serve German universities and research institutions to carry out research that requires extremely high CPU performance, high bandwidths to and from memory, to remote nodes and to disks, large main memory, and large disk storage capacity.

In its concept for the HLRB II, LRZ starts out from the assumption that the realization of coarse granular parallelization is easier and more efficient with a moderate number of high-performance nodes than with a large number of low-performance nodes. Therefore, MPP-like systems with many thousands of nodes with a relatively low individual performance - this includes workstation or PC clusters - will not be accepted for the planned system.

Additionally, the experiences with the current supercomputer show that many users utilize the hybrid programming model with MPI between the compute nodes and autoparallelization or OpenMP within the nodes.

Examples of computers that have the required characteristics include clusters with high-performance SMP-nodes (with 8 or more CPUs per node) or clusters of parallel vector processors (PVP). Many of the programs intended for execution on the new system, but not all, contain fine and medium grained parallelism and are also vectorizable. Therefore the system should be capable of efficiently processing programs via automatic fine and automatic medium grained parallelization within the SMP node and optionally via vectorization or vector-like techniques.

As mentioned before, the message passing paradigm will be used for coarse grained parallelism. As a consequence, the system must be equipped with a very efficient internal interconnect as well as an MPI implementation that makes optimal use of the hardware for communication between the nodes.

In order to achieve the highest possible level of performance, it will be necessary to exploit parallelism at all levels of a program, i.e., coarse grained parallelism at the level of domain decomposition, parallelism at the level of outer loops, and optionally vector-like parallelism at the level of inner loops. Thus, the objective of the HLRB II concept is to provide an environment offering a connection between coarse grained parallelization by domain decomposition, inevitable to achieve high performance, and the other two parallelization levels.

The following programming paradigms for parallelization should be available:

- Pure MPI Mode: message passing (+ optionally vectorization)
- Hybrid/Mixed Mode: message passing + shared memory (auto)parallelization/OpenMP (+ optionally vectorization).

In case that the performance of a single SMP node is powerful enough, the

- Pure Shared Memory Mode: (auto)parallelization/OpenMP (+ optionally vectorization)

is welcome by many users as an easy-to-use method. However, because of the generally limited scaling behavior of this mode, evaluation of the system is primarily based on the first two modes.

Experience and theoretical considerations show that for achieving optimal performance with the hybrid mode, a fully thread-safe implementation of MPI is essential.

The software necessary and commonly used for scientific and technical high-performance computing must be available (a UNIX or Linux operating system suitable for supercomputing centers, highly efficient compilers, libraries, tools, ISV applications).

Together with the compute nodes, the vendor must offer a sufficiently large and efficient mass storage subsystem for user data. The user storage sould be subdivided into two parts:

- a huge SAN/DAS part (Storage Area Network or Direct Attached Storage) which should contain the large user results files and temporary data. For these data high bandwidth and moderate access latencies are required.
- a NAS part (Network Attached Storage) for permanent user data, such as source files and input data. The size of the individual files will be moderate but the number of files will be high. Access latencies should be low, while bandwidth for the NAS part should be moderate. The NAS part should also be accessible from outside the HLRB II.

In addition to that, disks for the operating system, paging and/or swapping, etc. may be required.

Furthermore, the vendor should offer an efficient external communication network connection for access to the HLRB II from the German research network.

The expected job profile on the HRLB II requires, simultaneously, high computing performance, a large main memory with fast access characteristics, an efficient interconnect, and high I/O throughput. Therefore, it is most important that the offered supercomputer provides a balanced configuration of hardware and software.

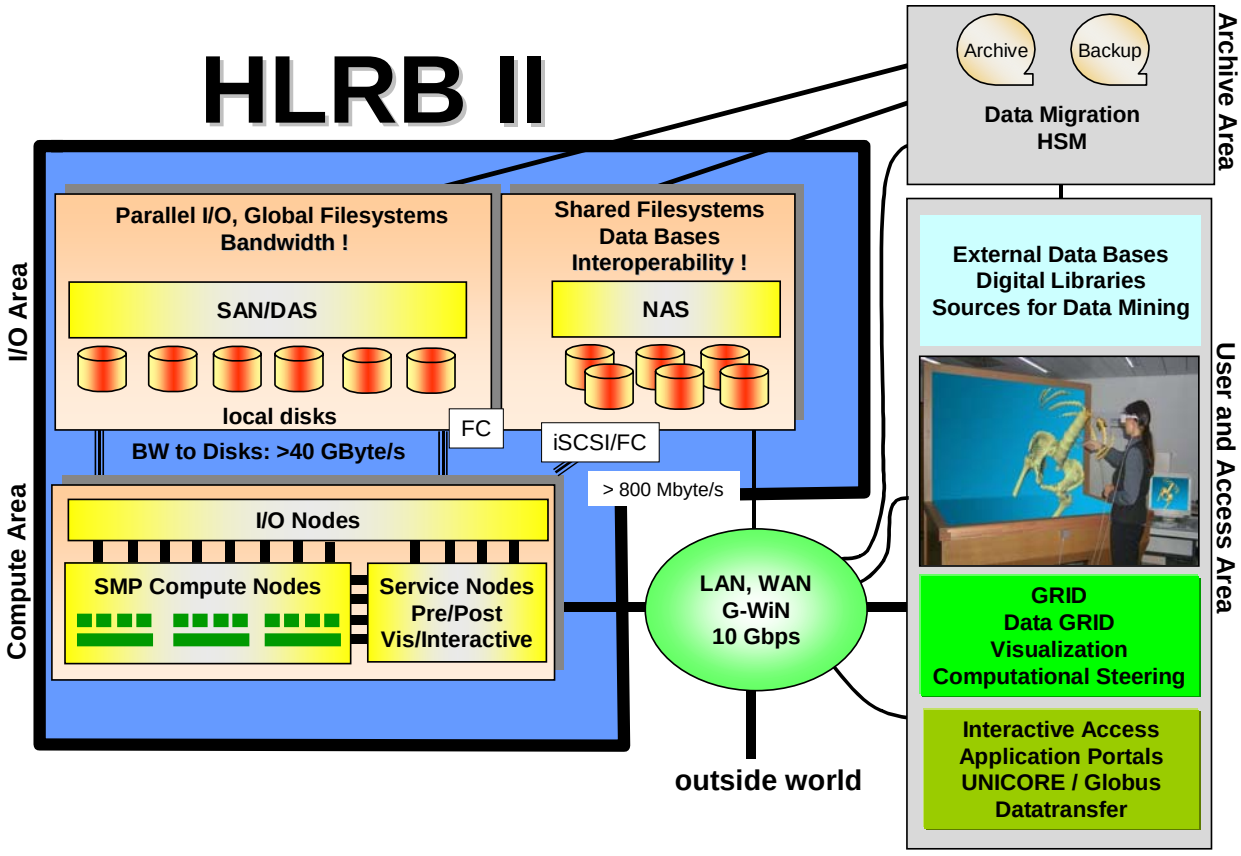
The installation of the HLRB II shall be accomplished in two phases:

- In Phase 1, starting at the end of 2005, 50 percent of the system resources should be installed.
- In Phase 2, starting in mid 2007, the rest of the system shall be installed.

Due to the fact that programs will have long run times, the offered HLRB II has to ensure high stability under permanent load with changing usage profiles.

Even in the first installation phase, the intended system should have the appropriate resources to achieve one of the upper positions among supercomputers worldwide.

The general concept of the systen is depicted in the following Figure. Only the area surrounded by the broad black line is subject of the present procurement.



1.2 Categorization of the requirements for the HLRB II

The detailed list of requirements given below for the HLRB II are categorized within this document by three classes which are marked by colored boxes:

M	Requirements within red boxes and marked by "M" are considered **mandatory**. It will constitute an exclusion criterion if they cannot be satisfied by the offered system.[1]

All other requirements must be fulfilled in their meaning and intended spirit, ensuring the operability and usability of the HLRB II.

I	Requirements within yellow boxes and marked by "I" are considered **important** for the operation and usability of the system. ´ In case that one or more of these non-exclusionary requirements cannot be fulfilled by the vendor, alternative offers and proposals for alteration are explicitly allowed as long as they provide equivalent characteristics. The proposed alternatives must be explained in detail. However, excessive failure to satisfy the spirit of the requirements may also constitute an exclusion criterion, or these failures may lead to a qualitative devaluation in conformance with the evaluation procedure.

T	Requirements within green boxes and marked by "T" are considered **targets** for optimal usage of the system. If a vendor can fulfill these requirements this will lead to a qualitatively better ranking through the evaluation procedure.

Many requirements contain checkboxes.

[ ] an empty checkbox means that the requirement cannot be fulfilled

[X] a checkbox marked by "x" means that the requirement can be fulfilled.

1.3 Terminology

The following terms are used in this document:

- **Phase 1:** refers to goods and services delivered for the first installation phase.
- **Phase 2:** refers to the **additional** goods and services delivered for the second phase.
- **Combined Phase 1+2:** refers to aspects of goods and services which can be rendered by the combination of Phase 1 and Phase 2 (e.g., performance of an application which runs on the entire system).
- **kByte, MByte, GByte or TByte** in the context of memory or cache means 2^{10}, $2^{20,}$ 2^{30}, 2^{40} Byte, resp. This also applies for memory/cache related bandwidths.
- **kByte, MByte, GByte or TByte** in the context of disk means 10^{3}, $10^{6,}$ 10^{9}, 10^{12} Byte, resp. This also applies to disk related bandwidths.

1.4 Requirements with respect to the content of the tender

Please explain in your tender how the given requirements can be fulfilled. All items mentioned in the following chapters must be covered.

[1] cf. "Bewerbungs- und Vertragsbedingungen"

The answers should refer only to offered goods and services and their alternatives. Answers and explanations must be inserted in the appropriate sections of this document and should be as concise as possible. It is preferred that the vendor uses the provided fields which are labeled by:

Answers and annotations by the vendor:

[Insert text here]

Additionally, the table at the end of this document with the key hardware parameters of the system must be filled in for Phase 1 and where applicable and known for Phase 2 and/or for the combined Phases 1+2. The vendor is free to provide additional information at the end of any paragraph.

Questions that have not been answered or have been answered insufficiently will be considered as answered in the negative.

Any difference between installation Phase 1 and Phase 2 must be explained in this description. Any description that does not explicitly refer to one of the installation phases is considered valid for both phases.

Wherever appropriate and possible, there should be references to common standards of information technology. Bibliographical references, also to brochures and manuals, are desirable, but can only be considered as supplementary information, i.e., they cannot replace the required answers or explanations.

The vendor may hand in the specifications stipulated below in German or English or a mixture of both.

1.5 Additional material

The following material is referenced in this document

- Specification of the new computer building
- CD containing the floor plan of the computer room, the benchmarks, this document as MS Word and PDF

2 Requirements for the HLRB II

The following chapter stipulates the technical requirements which the offered system needs to fulfil.

In many cases, there are, besides the requirements, additional questions which have to be answered. These questions will help to clarify facts relevant for the operation and usage of the offered HLRB II. The evaluation of the answers can result in a higher or lower ranking of the tender if the appropriateness of the system for operation as a national supercomputer seems to be particularly advantageous or disadvantageous when compared to the average of all tenders.

Aspects which will be evaluated include, but are not necessarily limited to:

- whether the configuration is balanced and therefore appropriate for the expected usage, with respect to the relevant values of

  - CPUs, processors or nodes
  - main memory
  - mass storage
  - internal interconnect, and
  - connection to the external communication network
- the relative and absolute bandwidths and latencies of the memory hierarchy, of the interconnects, and of the I/O subsystem
- the size of the shared memory of a node
- the extend of inhomogeneity between Phase1 and Phase2
- the flexibility in job administration and management, and verification of usage within the ambit of the explained aims of operation
- the possibilities provided by the software to ensure high quality of usability, a well-balanced load distribution, and operational security for the expected application profile
- the expected stability of hard- and software and downtimes caused by error recovery or debugging
- the expected downtimes caused by software maintenance and upgrades of the operating system
- the quality and availability of compilers, debuggers, test aids, and tools for performance analysis
- the extent and availability of means for control and evaluation of usage patterns
- the extent and availability of optimized scientific libraries and applications
- the expected improvement of usage due to the quality of documentation
- the support of general HPC activities by the vendor
- quality of vendor support for the operation of the HLRB II

2.1 Installation requirements

The description of installation requirements ensures that the offered system is physically installable and operable. The specifications must allow a reliable prediction of air conditioning, cooling demands, and energy consumption.

2.1.1 Proposal for installation

M 1:	A technical proposal for the installation for both phases including a detailed arrangement plan must be submitted.

Answers and annotations by the vendor:

[Insert text here]

2.1.2 Date of Installation

M 2:	Detailed information about: - the earliest installation date of Phase 1 - the earliest installation date of Phase 2 - the time needed to complete the installations - the length of interruption of service needed for the upgrade to Phase 2 must be submitted.

Answers and annotations by the vendor:

[Insert text here]

2.1.3 Delivery to the place of installation

The offered system shall be installed in the designated computer room of the new LRZ building. The freight elevator has clearance dimensions of

- width x height x depth = 1,55 m x 2.25 m x 2.25 m
- and a maximum payload of 2000 kg (~ 20 kN).

I 1:	The transportation of the offered system to the place of installation should be feasible under the above mentioned conditions Check here if the requirement can be fulfilled: **[ ]**

2.1.4 Installation of the system

An area in the new computer room of LRZ, which is specified in the floor plan included with the tender documents, is reserved for the system to be offered. The computer room is free of pillars and has the following dimensions and false floor specifications:

- width x depth x height = 28 m x 21 m x 8 m (= **588 m^2 x 8 m**)
- false floor with a maximum area load of **1500 kg per m^2** (~ 15 kN/m^2), a maximum "*point load*" of **500 kg** and a height of **1.8 m**

M 3:	Both installation phases of the offered system must be installable in the designated computer room of LRZ. Check here if the requirement can be fulfilled: **[ ]**

M 4:	The dimensions (i.e., foot print), maintenance areas, and weights of the devices must be submitted and they must be within specifications for the computer room. Check here if the requirement can be fulfilled: **[ ]**

M 5:	By checking the box, the vendor declares that the complete installation process and preparation for operation of the HLRB II (for both installation phases) is incumbent on the vendor[2]. Check here if the requirement can be fulfilled: **[ ]**

2.1.5 Power supply and conformity with electrical standards

LRZ provides power supply of 50 Hz alternating current, with 230 V single-phase and 400 V three-phase voltages, as commonly available in Germany. The planning and installation of electrical power and cooling equipment needed to run both phases will be performed during 2005.

I 2:	The total electrical power consumption of the combined Phases 1+2 of the HLRB II should not exceed 1800 kVA.[3] Check here if the requirement can be fulfilled: **[ ]**

I 3:	The offered system configuration for the HLRB II in both installation phases can be electrically connected and operated. Check here if the requirement can be fulfilled: **[ ]**

If other electrical power characteristics should be needed, the required frequency and voltage transformers must be included in the tender. The requirements for their installation (space demand, environmental conditions, and maximum distance to the HLRB II) must be specified in detail. The installation of such components is part of the installation of the HLRB II and cannot be charged separately.

M 6:	The characteristics of the power supply (e.g., voltage, number of phases, frequency) and the expected and peak energy consumption (in kVA) must be specified. The appropriate fields in the table at the end of this document must be filled in.

The *EC Directive on Electromagnetic Compatibility of Devices* is the guideline for all manufacturers, importers, and distributors of electric and electronic devices in the European Union as far as development, production and distribution of devices, systems and equipment are concerned. In addition, the *Low-Voltage Directives 73/23/EEC* and the *Act about Product Liability* must be considered.

These regulations and the corresponding directives and standards are EU stipulations that have been adopted by German Law.

By attaching a "CE marking" on a device, the manufacturer gives a legally binding declaration that the device conforms with all EMC relevant directives and standards of the EU and with the regulations and laws.

[2] LRZ only lends the necessary technical support.

[3] cf. cover letter ("Anschreiben")

M 7:	The devices must conform to German and European directives, i.e., they need to pass tests for the EC Directive on Electromagnetic Compatibility 89/336/EEC, Category A of limitation of interfering radiation (see EN55022-1998) and the above mentioned Low-Voltage Directives 73/23/EEC. All necessary documentation will be submitted to LRZ before delivery of the system. Check here if the requirement can be fulfilled: **[ ]**

T 1:	Classification in Category B of limitation of interfering radiation and CE-marking are desired. Check here if the requirement can be fulfilled: **[ ]**

2.1.6 Environmental conditions

M 8:	The required environmental conditions (e.g., ambient temperature, humidity, dust free conditions, etc.) for the main system and the peripheral devices must be specified. The appropriate fields in the table at the end of this document must be filled in.

M 9:	The type of cooling system for all devices (air-cooling, water-cooling, etc.) and details about the cooling system (e.g., in- and outlet temperature, temperature variation tolerance, purity requirements, etc.) must be specified. The appropriate fields in the table at the end of this document must be filled in.

M 10:	The heat emission of all devices of the offered system, broken down into air and water cooling, i.e., the maximum values and estimated values for permanent load, must be specified. The appropriate fields in the table at the end of this document must be filled in.

I 4:	The total heat emission for the combined Phases 1+2 should not exceed 1800 kW.[4] Check here if the requirement can be fulfilled: **[ ]**

2.1.7 Fire detection and protection

In case of fire, the computer room for the HLRB II can be flooded with inert gas. However, doing this is very expensive. Therefore fire protection, detection and/or extinction devices within the system are desired.

T 2:	Internal fire detection devices are provided. Check here if the requirement can be fulfilled: **[ ]**

T 3:	Internal fire protection devices are provided. Check here if the requirement can be fulfilled: **[ ]**

T 4:	Fire extinguishing agents which do not harm the system are provided. Check here if the requirement can be fulfilled: **[ ]**

If provided, describe the means of fire protection/detection/extinction in detail.

[4] cf. cover letter ("Anschreiben")

List the recommended fire extinguishing agents.

Answers and annotations by the vendor:

[Insert text here]

2.2 Hardware of the compute nodes

2.2.1 Compute node architecture

M 11: A detailed description of the compute node architecture (node architecture, processor architecture, etc.) is required.

The appropriate fields in the table at the end of this document must be filled in.

Additionally, the specifications of all components must be explained in detail and the following items must be considered:

- general description of architecture
- technology
- number of offered compute nodes, number of processors and size of main memory
- clock rates, number of floating and fixed point operations per clock (separately for scalar and/or vector)
- number of registers (separately for floating point, integer and general purpose operations, and for scalar and/or vector operations)
- number of execution units (e.g., arithmetic and general purpose units)
- if vector units are available, describe which data types are vectorized

Answers and annotations by the vendor:

[Insert text here]

2.2.2 Performance of the SMP compute nodes

According to the target group of applications for the HLRB II only nodes with high individual performance will be considered.

The offered compute nodes must achieve a minimum compute performance (optionally obtained by auto-vectorization and/or auto-parallelization and/or OpenMP directives) which is described in the benchmarking chapter (see section 4). Therefore:

I 5: All compute nodes should consist of SMP nodes with 8 or more CPUs.

Check here if the requirement can be fulfilled: **[ ]**

If you cannot fulfill this requirement, please explain why the offered nodes are suitable for the outlined designated usage of the system.

Answers and annotations by the vendor:

[Insert text here]

2.2.3 Extent of inhomogeneity of installation Phases 1 and 2

The HLRB II may be inhomogeneous in the two installation phases, e.g., processors of different clock frequency may be used for Phase 1 and Phase 2 and/or shared memory nodes with different numbers of CPUs may be used.

The transfer of applications between Phase 1 and 2 must be easily and readily possible. The intention of this is not to limit the option of having individual tuning and compiler parameters for a particular phase, but to make it possible to run a suitably generated executable on any of the two phases of the system.

In any case, the second phase installation must not result in an essential change of the programming paradigm, i.e., programs that achieve their computing performance by automatic parallelization and/or automatic vectorization, must run with a comparable efficiency and without major alterations at the program source level.

M 12:	It must be possible to run a suitably compiled executable on any compute node of the entire system. Check here if the requirement can be fulfilled: **[ ]**

M 13:	It must be possible to run a single program across the entire system. Check here if the requirement can be fulfilled: **[ ]**

M 14:	It must be possible to access all relevant user file systems on the NAS storage with equal efficiency from both phases (see 2.4.1). Check here if the requirement can be fulfilled: **[ ]**

I 6:	It should be possible to access all relevant parallel file systems on the SAN/DAS storage with approximately equal efficiency from both phases (see 2.4.1). Check here if the requirement can be fulfilled: **[ ]**

If it is not obvious in itself, please explain how these requirements are fulfilled.

Answers and annotations by the vendor:

[Insert text here]

2.2.4 Size of main memory

The offered main memory must have appropriate size and bandwidth in relation to the compute performance of the nodes.

In accordance to the observed user needs, the size of the memory available for user programs should satisfy the following criteria:

M 15:	The relative size of the main memory of a node must be at least 0.5 Byte per Flop/s theoretical peak performance. Check here if the requirement can be fulfilled: **[ ]**

M 16:	The minimum amount of memory per compute node must be at least 32 GBytes. Check here if the requirement can be fulfilled: **[ ]**

If the operating system or buffers for message passing etc. need a substantial fraction of the main memory, the vendor should disclose this information and take it into account in his offer, i.e. provided more than the required minimum.

However, if the vendor offers a substantially larger memory than required, this will not automatically be honored by LRZ's evaluation criteria. Therefore the vendor should explain the rationale behind this overcommitment.

Answers and annotations by the vendor:

[Insert text here]

2.2.5 Bandwidth of main memory

I 7:	The theoretical bandwidth of a node should exceed the value of 1 Byte/s per Flop/s theoretical peak performance. Check here if the requirement can be fulfilled: **[ ]**

However, the required bandwidth for many scientific applications is well above 3 Byte/s theoretical bandwidth per theoretical Flop/s. Therefore substantially larger bandwidth to memory than the required one will be positively honored by the LRZ evaluation scheme.

T 5:	It is desired that the relative bandwidth of a node exceed 3 Byte/s theoretical bandwidth per Flop/s theoretical peak performance. Check here if the requirement can be fulfilled: **[ ]**

The need for bandwidth to the local memory can be partially compensated by very large caches, therefore:

T 6:	For systems with low memory bandwidth, it is desired that the size of the largest cache exceeds 6 MB per CPU. Check here if the requirement can be fulfilled: **[ ]**

2.2.6 Memory and cache hierarchy

M 17:	The vendor must disclose detailed information about the memory subsystem and, if applicable, of the cache hierarchy of the compute nodes, i.e., - latencies - bandwidths - size and associativity of caches and/or vector registers - access patterns (cache line sizes, maximum number of outstanding prefetches, memory interleaving, etc.) - TLB size (entries) and latency The appropriate fields in the table at the end of this document must be filled in.

Additionally, the following information should be disclosed:

- the influence of a stride different from one on memory access time
- possibilities of memory conflicts and the delays they will cause (locally, on the node, and over the whole network).
- special hardware characteristics that support operations on sparse matrices or indirect addressing (Gather/Scatter or similar)
- number of processors that can access the shared main memory without significant delay, if each processor requires the maximum memory bandwidth with respect to its memory bus

- the structure of address space of the system must be indicated.
- the cache coherency should be described (i.e., cc-NUMA[5] or non-cc-NUMA).
- the policies for data placement (e.g., first touch, page migration)

Answers and annotations by the vendor:

[Insert text here]

2.2.7 Memory protection and error detection

M 18:	Mechanisms for error detection and correction in main memory must be provided. Check here if the requirement can be fulfilled: **[ ]**

T 7:	Extended memory protection mechanisms (like Chipkill) are desired. Check here if the requirement can be fulfilled: **[ ]**

Please describe these mechanisms.

Answers and annotations by the vendor:

[Insert text here]

2.2.8 Hardware support of parallelization

T 8:	Fast hardware synchronization to support SMP-parallelization within a node is desired. Check here if the requirement can be fulfilled: **[ ]**

If available, please describe.

What are the available hardware characteristics to support parallelization between nodes (e.g., RDMA, global address space etc.) ?

Answers and annotations by the vendor:

[Insert text here]

2.3 Communication network

2.3.1 Hardware of internal interconnect

M 19:	A detailed description of the internal interconnect within Phase1 and Phase 2, as well as of the interconnect between the two phases must be given. The appropriate fields in the table at the end of this document must be filled in.

The following items should be disclosed:

- general description of network structure and node connections
- technology

[5] ccNUMA = cache coherent Non Uniform Memory Access

- topology
- bandwidth and latency (where appropriate, taking into account distance between nodes)
- scalability with number of compute nodes
- performance of all involved single components
- limitations (e.g., message length, number of outstanding messages)
- maximum time for a broadcast to reach all nodes

Answers and annotations by the vendor:

[Insert text here]

2.3.2 Interconnect within Phase 1 or within Phase 2

The performance of the interconnect within Phase 1 and within Phase 2, respectively, must be in appropriate relation to the compute performance of the nodes.

I 8: The interconnect balance

= **aggregate MPI link bandwidth (Byte/s) of a node**
(as defined by the MPI benchmark in 4.1.2)
divided by
the weighted performance of the vector triad (Flop/s) of a node
(as defined by Case 2 of the RINF1 benchmark in 4.3.1)

should exceed **0.4 Byte/s/Flop/s**.

Check here if the requirement can be fulfilled: **[ ]**

T 9: It is desired that the interconnect balance of one node (as defined above) exceeds 0.8 Bytes/s per Flop/s

Check here if the requirement can be fulfilled: **[ ]**

T 10: It is desired that the theoretical (peak) interconnect bandwidth of one node exceeds 5% of the theoretical memory bandwidth of that node.

Check here if the requirement can be fulfilled: **[ ]**

M 20: The MPI latency, between two arbitrary nodes within Phase 1 or between two arbitrary nodes within Phase 2 must not exceed 8 µs.

Check here if the requirement can be fulfilled: **[ ]**

T 11: It is desired that the MPI latency between two arbitrary nodes within any phase does not exceed 5 µs.

Check here if the requirement can be fulfilled: **[ ]**

2.3.3 Interconnect between Phase 1 and Phase 2

The interconnect between the two phases of HLRB II may be different from that within Phase 1 or Phase 2, respectively. Notwithstanding, starting a parallel program across phase boundaries must be possible in a transparent manner.

One important aspect for the interconnect between the two phases is to handle the I/O traffic. This defines a lower limit for the required bandwidth.

M 21:	The MPI bidirectional bisection bandwidth between Phase 1 and Phase 2 as defined by the benchmark in 4.1.4 must exceed 80 GByte/s (40 GByte/s in each direction). Check here if the requirement can be fulfilled: **[ ]**

I 9:	The MPI latency across the phase boundaries between two arbitrary nodes should not exceed 10 µs. Check here if the requirement can be fulfilled: **[ ]**

T 12:	It is desired that the interconnect between the two phases is of the same type and characteristics as within a phase. Check here if the requirement can be fulfilled: **[ ]**

2.3.4 Control network

Are there, besides the internal interconnect for data exchange in user programs, other internal networks (e.g., for system purposes)? If yes, please describe their structure and function.

Answers and annotations by the vendor:

[Insert text here]

2.3.5 Connection to external network

I 10:	10 Gigabit ethernet (10 GE) interfaces with TCP/IP offload functionality (TOE) or RDMA functionality should be delivered as communication interfaces for connecting the HLRB II to the outside world for both phases. Check here if the requirement can be fulfilled: **[ ]**

M 22:	At least two independent network connections to the external network using a standard ethernet frame size of 1500 Bytes must be provided for each phase. Check here if the requirement can be fulfilled: **[ ]**

M 23:	The external network connection of the system is considered to be balanced if it offers a measurable bandwidth of more than 600 MByte/s to the NAS filer for Phase 1 and 800 MByte/s for Phase 2 according to IO-Bench 2 (see section 4.2.2). Check here if the requirement can be fulfilled: **[ ]**

A detailed description of the available external 10 GE communication interfaces (or statements about their planned availability) with all relevant performance specifications (hardware and typical achievable values under TCP/IP protocol stack as well as port trunking support, cable type and maximum cable length) is required.

Answers and annotations by the vendor:

[Insert text here]

2.4 Disk storage

2.4.1 User disk storage

Two different types of user mass storage have to be delivered with HLRB II:

- a parallel, high-performance file system with very high aggregate bandwidth, which can be accessed by every node (Storage Area Network/Direct Attached Storage, SAN/DAS)
- a Network Attached Storage (NAS) connected over Gigabit Ethernet or 10 Gigabit Ethernet and **NFSv3** or higher, to ensure a high interoperability with applications outside the HLRB II.

M 24:	All user mass storage has to be RAID-protected, using at least RAID levels RAID1, RAID3, RAID4, RAID5 or higher. One or more spare disks with automatic take-over, redundant controllers with mirrored or deactivated controller caches must be provided. Check here if the requirement can be fulfilled: **[ ]**

M 25:	A net value of at least 300 TByte[6] SAN/DAS mass storage has to be delivered for each phase. Check here if the requirement can be fulfilled: **[ ]**

I 11:	It should be possible that a single (parallel) file system serves this storage to the users for each of the phases. Check here if the requirement can be fulfilled: **[ ]**

I 12:	The storage systems should support scheduled media scans with configurable priority and manual or automatic cloning of disk drives showing an increasing number of bad blocks. Check here if the requirement can be fulfilled: **[ ]**

M 26:	A net value of at least 40 TByte[7] of Network Attached Storage must be delivered with Phase 1 and additional 20 TByte[8] of Network Attached Storage have to be delivered with Phase 2 of HLRB II. For this type of user storage a disk technology built for high I/O rates has to be used, i.e., native SCSI, Fibre Channel or Serial Attached SCSI disks (SAS). Check here if the requirement can be fulfilled: **[ ]**

I 13:	It should be possible that all NAS capacity is accessible in a single file system. Check here if the requirement can be fulfilled: **[ ]**

I 14:	The Network Attached Storage should be implemented as a **highly available** stand-alone solution, e.g., it should be possible to access all data on this storage from outside independent from the status of the HLRB II.

[6] **In this context 1 TByte is defined as 10^{12} Bytes**

[7] **In this context 1 TByte is defined as 10^{12} Bytes**

[8] **In this context 1 TByte is defined as 10^{12} Bytes**

	In addition it should be possible to access all NAS data from outside using a standard ethernet frame size of 1500 bytes. Check here if the requirement can be fulfilled: **[ ]**

I 15:	The NAS solution should provide an automatic as well as a manual **file system snapshot** functionality and should support an automatic **replication** (and different levels of synchronicity) of the snapshots over SAN and/or LAN to one or more other network filer. Check here if the requirement can be fulfilled: **[ ]**

I 16	The NAS solution should support data backup using NDMP together with Tivoli Storage Management (TSM) servers (see also 2.9.2). Check here if the requirement can be fulfilled: **[ ]**

M 27:	The offered I/O solutions and disk technologies must be described in detail.

The description should contain information about

- the expected MTBF of the single devices
- capacity
- number of drives
- redundancy
- used channel protocol
- peak, average, and sustainable transfer rates and access times
- how disk errors will be reported by the operating system.
- a detailed description of the SAN/DAS solution containing all relevant features, such as hardware specifications, reliability, management software and interfaces, functionalities for protection against data loss (e.g., number of spare disks, supported RAID levels, snapshots and replication).
- a detailed description of the NAS solution containing all relevant features, such as hardware specifications, reliability, management software and interfaces, functionalities for protection against data loss (e.g., number of spare disks, supported RAID levels (double parity), snapshots, replication, fast backup and restore of data).

Answers and annotations by the vendor:

[Insert text here]

2.4.2 Disk storage containing the operating system, swap and/or paging space of the nodes

M 28:	All disk storage containing the operating system, swap and/or paging space of the nodes has to be RAID-protected, using RAID-level RAID1 or higher. Check here if the requirement can be fulfilled: **[ ]**

M 29:	The offered disk capacity and technology must be described.

Answers and annotations by the vendor:

[Insert text here]

2.4.3 I/O nodes

M 30: I/O nodes (or nodes specially equipped for I/O) and their input/output channels must be described.

The description must include:

- general description of architecture, including performance specifications
- technology
- number of offered nodes, their equipment with processors and main memory
- procedure for adding I/O nodes
- maximum possible number of I/O nodes.

Answers and annotations by the vendor:

[Insert text here]

2.4.4 Management tools for the disk subsystem

T 13: Graphical management tools for the disk system are desired.

Check here if the requirement can be fulfilled: **[ ]**

If available, please describe them.

Answers and annotations by the vendor:

[Insert text here]

2.5 Hardware faults

2.5.1 Detection of hardware faults

M 31: Facilities for the online detection of hardware errors (e.g., faulty memory modules, processors, fans, network links, switches) must be available.

Check here if the requirement can be fulfilled: **[ ]**

Particularly describe:

- the facilities for diagnosis and error correction
- the process of diagnosis and error correction
- whether there are any spare nodes or spare processors and what their functions are
- whether faulty nodes or processors can be taken out of service without interruption of operation and, if necessary, be substituted by spare nodes or spare processors
- whether error correction can be carried out online

Answers and annotations by the vendor:

[Insert text here]

2.5.2 Mean time between failures

M 32:	An estimate and rationale for the expected mean time between failures (MTBF) of the overall system and its major components must be given.

Answers and annotations by the vendor:

[Insert text here]

2.6 Operating system

If the operating system (partially) runs on a front-end computer the following requirements must be explained in the tender for the supercomputer itself as well as for the front-end computer, wherever this is reasonable.

System components for resource administration and batch administration as well as other components that are essential for system operation are considered part of the operating system even if, formally, the manufacturer (or another vendor) should offer them separately.

2.6.1 64-bit operating system

M 33:	The system must be delivered with a 64-bit operating system using a 64-bit kernel. Check here if the requirement can be fulfilled: **[ ]**

Please describe the 64-bit implementation details.

Answers and annotations by the vendor:

[Insert text here]

2.6.2 Standards

M 34:	The operating system must be complete, stable and appropriate for production usage in a supercomputing center, i.e., it must allow flexible administration and control of interactive and batch mode. Check here if the requirement can be fulfilled: **[ ]**

I 17:	The operating system should be based on UNIX or Linux and should be compatible with the X/Open Standard POSIX 1003 (ISO/IEC 9945). Check here if the requirement can be fulfilled: **[ ]**

Answers and annotations by the vendor:

[Insert text here]

2.6.3 Stability

Since the HLRB II will usually run 24 hours a day, high system stability and easy recoverability is of great importance. Special attention has to be paid to the problem of interruptions due to hardware failures.

M 35:	In case of a failure of a compute or I/O node a seamless degradation of the system must be warranted.
	Check here if the requirement can be fulfilled: **[ ]**

Please describe in detail how the operation of the system with lower performance in case of failure of a component (e.g., a CPU, a compute node, an I/O node, a part of the main memory, a part of the disks) will be guaranteed.

Please describe how the repaired components will be returned to operation. In which cases is this possible without interruption of normal operations, in which cases can an interruption not be avoided?

Describe which components may cause an interruption of the complete system.

Answers and annotations by the vendor:

[Insert text here]

2.6.4 Checkpointing

As a provision against unplanned interrupts the possibility of checkpoint/restart is desired.

T 14:	The possibility of user initiated checkpoint/restart is desired.
	Check here if the requirement can be fulfilled: **[ ]**

T 15:	The possibility of system initiated checkpoint/restart is desired.
	Check here if the requirement can be fulfilled: **[ ]**

If available, describe the possibilities of checkpoint/restart (for sequential/parallel programs); describe the possibilities of system wide, preventive checkpointing, allowing to stop the normal (batch) operation, carry out maintenance tasks and afterwards return to normal operation.

Answers and annotations by the vendor:

[Insert text here]

2.6.5 Job freeze

Freezing a job means interrupting execution of a job and saving the job and its environment onto disk, so that all resources used by this job are released and can be used by other jobs. It is especially useful for short term scheduling of jobs on a filled machine.

T 16:	It is desired to have job freeze capabilities.
	Check here if the requirement can be fulfilled: **[ ]**

If available, describe the possibilities of job freeze.

Answers and annotations by the vendor:

[Insert text here]

2.6.6 Workload management (WLM)

T 17:	The operating system should have the ability to efficiently manage different workloads of the system, i.e., dynamically grant resources to system tasks depending on a classification scheme decided on by the system administrator. Check here if the requirement can be fulfilled: **[ ]**

Please describe the possibilities of the workload management of the operating system and the interaction of WLM with the batch scheduling system.

Can processes be allocated to certain processors at start time?

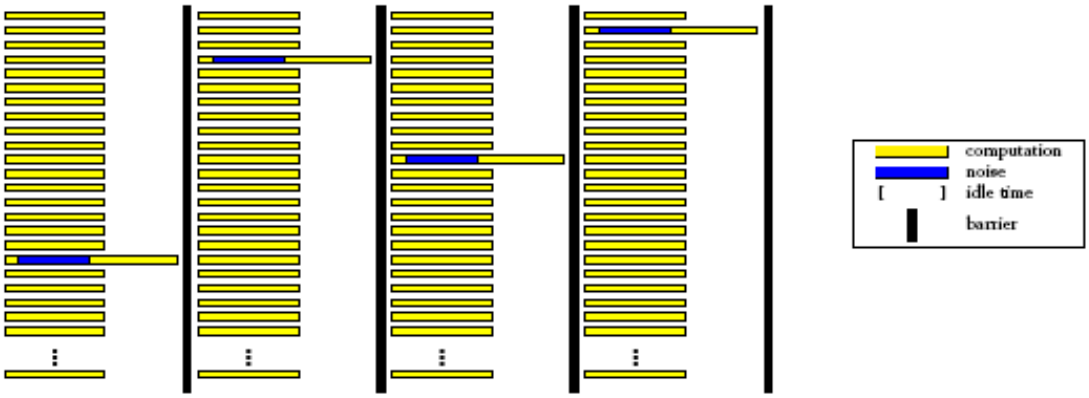
Can processes be bound to certain processors at run time?

Answers and annotations by the vendor:

[Insert text here]

2.6.7 Operating System induced scheduling noise

On many computers, system activities can run without interfering with the application as long as there is a spare processor available in each node to absorb them. When there is no spare processor, a processor may be temporarily taken from the application to handle the system activity. Doing so may introduce performance variability, which we refer to as “noise”. The following figure provides an illustration of the potential effects of these delays on applications that are fine-grained and bulk-synchronous. In such a case, a delay in a single process slows down the whole application considerably, particularly in the case of hundreds of processors.[9]



T 18:	Provisions to minimize system activities that may disturb user applications ("scheduling noise ") are desired. Check here if the requirement can be fulfilled: **[ ]**

If available, please describe these provisions.

Answers and annotations by the vendor:

[Insert text here]

[9] For detailed discussion see: http://www.sc-conference.org/sc2003/paperpdfs/pap301.pdf

2.6.8 Gang scheduling

T 19:	Mechanisms like gang scheduling are desired. Check here if the requirement can be fulfilled:	**[ ]**

If available, please describe details.

Answers and annotations by the vendor:

[Insert text here]

2.6.9 Support for large pages

I 18:	The operating system should support large page sizes (i.e., page sizes much larger than 4 kByte). Check here if the requirement can be fulfilled:	**[ ]**

I 19:	It should be possible to dynamically (on a per process or job basis) alter the number of large pages available on a node, depending on user demands. In any case, no reboot of the system should be required to accomplish this task. Check here if the requirement can be fulfilled:	**[ ]**

Please describe the corresponding possibilities of your system in detail.

Answers and annotations by the vendor:

[Insert text here]

2.7 Batch processing

2.7.1 Batch scheduling system

M 36:	The batch scheduling system must be able to classify batch jobs into different job classes. It must be possible to charge for used resources on a per user and on a per job basis, depending on the chosen job policy. Check here if the requirement can be fulfilled:	**[ ]**

I 20:	The batch scheduling system should be capable of starting jobs in a reasonably short time without too much loss from empty nodes (e.g., through backfill scheduling). Check here if the requirement can be fulfilled:	**[ ]**

M 37:	The batch scheduling system must be capable of supporting Grid software, particularly Globus and UNICORE. Check here if the requirement can be fulfilled:	**[ ]**

A suggestion for the integration of Grid software like Globus and UNICORE into the offered scheduling system has to be given.

Answers and annotations by the vendor:

[Insert text here]

I 21:	The scheduling system should support the allocation of CPUs for batch mode and interactive mode, depending on demand. Check here if the requirement can be fulfilled: **[ ]**

Please explain, how reasonable response times for interactive mode can be ensured under conditions of full batch load? Indicate precautions to avoid that typical batch workloads are carried out in interactive mode (e.g., limitation of resources for interactive jobs).

It has to be explained how the load balancing system will avoid that single processors remain unused.

Answers and annotations by the vendor:

[Insert text here]

I 22:	The application of the Maui scheduler should be supported by the scheduling system **or alternatively** the scheduling system should be capable of satisfying preemption needs for computational steering (preemption scheduler). Check here if the requirement can be fulfilled: **[ ]**

I 23:	The scheduling system should be capable of freezing serial and parallel jobs. Check here if the requirement can be fulfilled: **[ ]**

The possibilities and restrictions (if any) of the offered batch scheduling system have to be described, e.g.:

- Does the batch system support the concept of a job or does it, for example, provide accounting data only for single processes, without indicating their connection to a certain job? How can job-wise accounting be implemented in such cases?
- Can nodes or CPUs be dynamically post-allocated and deallocated (which would be useful whenever a program requires a high number of nodes or CPUs only for a small portion of its run time and for the rest of the time can do with less)? Do all involved components support the deallocation of nodes or CPUs by a user and allow their reallocation to another user?
- Is it possible that a user program, instead of requiring a fixed number of nodes or CPUs, indicates an interval (from a minimum to a maximum number) and will be scheduled as soon as the required minimum number of nodes or CPUs is available?
- Please describe in particular the possibilities the LRZ has to implement a job initialization of its own.
- Can the batch scheduling system be configured in a way that the system prologues are executed for batch jobs?

Answers and annotations by the vendor:

[Insert text here]

2.7.2 Job surveys

M 38: All jobs must be clearly and permanently identifiable and addressable by a system administrator.

A system administrator must be able to access all job characteristics (user ID, user job name, service and resource job category, submitting system, currently processed job steps, already used resources, etc.) by fast and easy means.

Check here if the requirement can be fulfilled: **[ ]**

Job control

M 39: System administrators must be able to delete, assign priorities, or defer any job at any processing level.

Check here if the requirement can be fulfilled: **[ ]**

2.7.3 Coexistence of batch and interactive mode

I 24: It should be possible to separately allocate certain privileges and limits for resource demands (e.g., CPU time, main memory, number of allocatable processors) to users and user groups for interactive and batch operation.

Check here if the requirement can be fulfilled: **[ ]**

I 25: Within the existing resource limitations, it should be possible to run parallel interactive jobs.

Check here if the requirement can be fulfilled: **[ ]**

I 26: Users should be able to log on (with ssh) to the batch nodes presently running their programs. Access to other batch nodes should be disabled.

Check here if the requirement can be fulfilled: **[ ]**

Please explain how interactive program development (debugging, profiling, tuning) of parallel programs can be performed under these conditions.

Answers and annotations by the vendor:

[Insert text here]

2.7.4 Co-scheduling and resource reservation

Co-scheduling is the ability to schedule various resources for usage at the same time. Co-scheduling, e.g., for simulation and interactive visualization, is important in the context of Metacomputing and grid computing. Co-scheduling may happen on the level of the grid middleware; it has to be supported, e.g., by advance resource reservation capability of the computer's batch system. This is currently difficult, as batch access to large-scale resources is not guaranteed at a specified time. Job prioritization and migration mechanisms may be other means to support co-scheduling.

T 20: Means for co-scheduling and/or resource reservation are desired.

Check here if the requirement can be fulfilled: **[ ]**

If available, please describe the details.

Answers and annotations by the vendor:

[Insert text here]

2.8 Administration

2.8.1 User administration

M 40:	Secure user administration must be possible, e.g. the encrypted password files must be protected against being read by unprivileged users. Check here if the requirement can be fulfilled: **[ ]**

I 27:	The user administration should allow fine grained settings on a per user and/or per node basis (e.g., whether the user may log in to a node or only may run batch jobs on it). This includes the possibility to determine user quotas and user restrictions with respect to computing time, main memory demand, number of allocatable compute nodes and mass storage as well as logging of used resources for administration and charging of user accounts. Check here if the requirement can be fulfilled: **[ ]**

I 28:	It should be possible to carry out all tasks of user administration via scriptable interfaces including setting of new passwords without interactive editing of files by the system administrator. Check here if the requirement can be fulfilled: **[ ]**

2.8.2 User validation

M 41	The operating system must provide reliable mechanisms to restrict any access to the system without successful validation. Check here if the requirement can be fulfilled: **[ ]**

I 29:	The operating system of HLRB II should support Pluggable Authentication Modules (PAM) for user validation. Check here if the requirement can be fulfilled: **[ ]**

I 30:	Pluggable authentication modules for LDAP and Kerberos5 should be provided. Check here if the requirement can be fulfilled: **[ ]**

T 21:	The availability of other secure authentication mechanisms is desirable. Please describe them.

Answers and annotations by the vendor:

[Insert text here]

2.8.3 System prologues

M 42:	A system prologue must exist for each login shell, the execution of which must be mandatory for every login and cannot be interrupted by the user, e.g., in order to control quotas or set the limits for resource demands.
	Check here if the requirement can be fulfilled: **[ ]**

M 43:	System administrators must be able to define a set of selectable shells and commands.
	Check here if the requirement can be fulfilled: **[ ]**

2.8.4 Security

M 44:	Bugs and security issues discovered during operation of the system must be eliminated reliably and with a reasonable response time.
	Check here if the requirement can be fulfilled: **[ ]**

T 22:	CSooperation with CERT is desired.
	Check here if the requirement can be fulfilled: **[ ]**

2.8.5 Secure Shell

M 45:	The UNIX standard commands rsh, rcp and rlogin (as well as rshd, rlogind, etc.) must not be available for users (or even necessary for the use of the HLRB II), but must be replaced by the products of the secure shell family (sshd, ssh, scp, slogin, etc.).
	Check here if the requirement can be fulfilled: **[ ]**

2.8.6 Configuration files

M 46:	All offered administrative programs have to provide secure and stable conditions of configuration files, e.g., they have to ensure that no files are left write-enabled for public access.
	Check here if the requirement can be fulfilled: **[ ]**

2.9 Data Handling

2.9.1 File systems

M 47	The maximum possible size for a single file in the parallel file system (SAN/DAS) must be well above 1 TByte.
	Check here if the requirement can be fulfilled: **[ ]**

I 31:	The maximum possible size for a single parallel file system (SAN/DAS) should be at least as large as the offered disk space.
	Check here if the requirement can be fulfilled: **[ ]**

I 32	All file systems delivered with HLRB II should support the journaling of meta- and/or user data or some equivalent mechanism. Check here if the requirements can be fulfilled: **[ ]**

Please describe the technical implementations of the journaling feature or equivalent mechanisms used by HLRB II file systems.

T 23	File system and/or consistency checkers for all file systems are desired. Check here if the requirements can be fulfilled: **[ ]**

Answers and annotations by the vendor:

[Insert text here]

2.9.2 Archive and backup

M 48:	A fully functional TSM (Tivoli Storage Management) client optimized for high transfer rates supporting LANless backup/archive must be included in the tender. The vendor must commit himself to deliver an update to the TSM client at most one year after a new release is available from IBM. Check here if the requirement can be fulfilled: **[ ]**

T 24:	A TSM client supporting serverless backup/archive is desired. Check here if the requirement can be fulfilled: **[ ]**

Please describe the possibilities of the TSM client.

Answers and annotations by the vendor:

[Insert text here]

2.9.3 Hierarchical storage management

T 25:	A hierarchical storage management solution in conjunction with TSM is desired. Check here if the requirement can be fulfilled: **[ ]**

If available, please describe implementation and functionality of the HSM solution.

Answers and annotations by the vendor:

[Insert text here]

2.9.4 AFS

T 26:	A fully functional AFS (Andrew File System) client is desired. Check here if the requirement can be fulfilled: **[ ]**

2.10 Monitoring

2.10.1 System monitoring

T 27:	An SNMP agent with Management Information Bases (standard MIB and a host resources MIB) for automatic system monitoring by means of network management systems is desired. Check here if the requirement can be fulfilled: **[ ]**

Please describe what possibilities of automated system monitoring your system supports.

Answers and annotations by the vendor:

[Insert text here]

2.10.2 Verification of system configuration

I 33:	Mechanisms for the system administrator should be available that allow the detection of divergences from a well-defined configuration (e.g., alterations of access rights or exchanged files or commands). Check here if the requirement can be fulfilled: **[ ]**

If available, please describe the mechanisms.

Answers and annotations by the vendor:

[Insert text here]

2.10.3 Log files

I 34:	Detailed and complete log files that store all security relevant incidents (e.g., every issued command and its context, such as parameters and options) should be available. Suitable interfaces for the processing of log files have to be provided. Check here if the requirement can be fulfilled: **[ ]**

I 35:	The format of log files should be uniform, informative and readable and these files should not be scattered in many single directories. Check here if the requirement can be fulfilled: **[ ]**

I 36:	Log files should not be created as files write-enabled for public. Check here if the requirement can be fulfilled: **[ ]**

Please describe the available tools and mechanisms.

Answers and annotations by the vendor:

[Insert text here]

2.10.4 Multilevel administration rights

T 28:	It is desired to have a multilevel administration with correspondingly leveled capabilities, e.g., for the system administrator, operators and common users. Check here if the requirement can be fulfilled: **[ ]**

If available, please describe the extent and functionality.

Answers and annotations by the vendor:

[Insert text here]

2.10.5 Status of operating system

I 37:	As far as system analysis is concerned, it should be possible to obtain current reports on the most important tables of the operating system at any time. Check here if the requirement can be fulfilled: **[ ]**

I 38:	Analysis tools interpreting the internal tables should be available and be supported by the vendor. Check here if the requirement can be fulfilled: **[ ]**

2.10.6 Operating system errors

M 49:	An error tracking and reporting mechanism in cases of operating system errors (e.g., system dumps) must be available and officially supported by the vendor. A description of the reporting and error correction procedure must be given. Check here if the requirement can be fulfilled: **[ ]**

Answers and annotations by the vendor:

[Insert text here]

M 50:	Adequate disk space for holding memory dumps etc. must be available. Check here if the requirement can be fulfilled: **[ ]**

2.10.7 Operator interface

T 29:	The existence of a GUI-based as well as a command-line operator interface is desired. Check here if the requirement can be fulfilled: **[ ]**

If any operator interface is available, describe its extent, particularly:

- What can operators do to control and supervise the system without the need of super user validations?
- For actions that usually require super user validations: How can operators carry them out?

Answers and annotations by the vendor:

[Insert text here]

2.10.8 Operation surveys

I 39:	Surveys with different level of detail should be available providing fast and easy access to important operation parameters, data, and statistics. Check here if the requirement can be fulfilled: **[ ]**

If such surveys are available, describe their extent.

Answers and annotations by the vendor:

[Insert text here]

2.11 System tuning and testing

2.11.1 Tuning of the operating system and performance monitoring

I 40:	Special tuning tools should be provided to allow the specific setting of system parameters: It should be possible to change system parameters without system interruption. Software monitors for the measurement of all important system characteristics (I/O behavior, disk access behavior, CPU load, memory load, paging rate, etc.) must be available with an easy-to-interpret output. Check here if the requirement can be fulfilled: **[ ]**

I 41:	The operating system should provide tools to extract performance information like number of floating point operations, number of integer operations, memory references, etc. per second from hardware performance counters. This information should be available to the system administrators on a per CPU basis without any impact on user codes and without the necessity of any specific changes to those codes. Check here if the requirement can be fulfilled: **[ ]**

I 42	It should be possible to monitor the interconnect of the system, e.g., determine the number and size of packets and the amount of data sent or received. Check here if the requirement can be fulfilled: **[ ]**

A detailed description of the tuning tools, software monitors, and performance analysis tools for the system administrator should be provided.

Answers and annotations by the vendor:

[Insert text here]

2.11.2 Test versions of software

I 43:	It should be possible to run different versions of compilers, linkers, libraries, applications, etc. alternatively or additionally to the standard programming environment. Check here if the requirement can be fulfilled: **[ ]**

2.12 System restarts and upgrades

2.12.1 System restarts

Please describe which kind of initial program starts the operating system and from which media a restart can be initiated?

M 51:	The time required for a complete system restart must be specified.

Answers and annotations by the vendor:

[Insert text here]

2.12.2 Modifications of the operating system

I 44	Modifications of all parts of the operating system (except the kernel) should be possible at any time, without interrupting the operation, immediately taking effect, and it should be possible to undo single modifications separately. Check here if the requirement can be fulfilled: **[ ]**

Please describe how releases and patches of the operating system will be delivered (medium?) and installed.

Answers and annotations by the vendor:

[Insert text here]

2.12.3 Operating system upgrade

I 45	The time required for an operating system upgrade or complete installation of a new operating system on all nodes should be well below 24 hours.	
	Check here if the requirement can be fulfilled:	**[ ]**

2.13 Software

Please indicate explicitly whether one of the products mentioned in this section can only be provided by third-party manufacturers.

2.13.1 Compilers

M 52:	The Fortran compiler must support a full implementation of the language specifications of Fortran 95 (ANSI X3J3/96-007).	
	Check here if the requirement can be fulfilled:	**[ ]**

M 53:	The C Compiler must at least support a full implementation of the standard ANSI/ISO 9899-1990 („C90“):	
	Check here if the requirement can be fulfilled:	**[ ]**

M 54:	The C++ Compiler must at least support a full implementation of the standard ANSI/ISO 14882-1998 („C++98“), including the C++ standard library.	
	Check here if the requirement can be fulfilled:	**[ ]**

M 55:	Compilers must support OpenMP, at least version 1.1.	
	Check here if the requirement can be fulfilled:	**[ ]**

M 56:	Interoperability between Fortran, C, and C++ must be provided.	
	Check here if the requirement can be fulfilled:	**[ ]**

M 57:	Compilers must support 64-bit mode.	
	Check here if the requirement can be fulfilled:	**[ ]**

I 46	OpenMP 2.0 for Fortran should be supported.	
	Check here if the requirement can be fulfilled:	**[ ]**

I 47	A port of a recent version of the GNU compiler collection should be available.	
	Check here if the requirement can be fulfilled:	**[ ]**

Please explain which versions are available.

Answers and annotations by the vendor:

[Insert text here]

I 48:	SMP auto-parallelizing and optionally auto-vectorizing compilers for Fortran C, and C++ are needed; furthermore, tools that support the users in parallelization, vectorization and performance analysis should be available. Check here if the requirement can be fulfilled:	**[ ]**

T 30:	Tools for automatic or user-guided conversion of programs into OpenMP programs are desired. Check here if the requirement can be fulfilled:	**[ ]**

T 31:	The C++ compiler should support the ISO Technical Corrigendum No.1, (TC1), i.e. the revision of the standard issued in 2003 ("C++03"), see http://www.open-std.org/JTC1/SC22/WG21/docs/cwg_active.html: Check here if the requirement can be fulfilled:	**[ ]**

T 32:	Full support for the Fortran 2003 standard is desired. Check here if the requirement can be fulfilled:	**[ ]**

If Fortran 2003 is not fully supported, please fill in the checkboxes in the following table indicating the level of support for Fortran 2003 features of your compiler. In this table the usual categorization for requirements applies.

Chapter[10] or Reference	Fortran 2003 Functionality	Categorization	Availability Phase 1	Availability Phase 2
ISO/IEC TR15581	ALLOCATABLE attribute on dummy arguments, function results and structure components	I	**[ ]**	**[ ]**
3.3, 3.4	Assignment to an allocatable array, transferring an allocation	I	**[ ]**	**[ ]**
3.9	VOLATILE	I	**[ ]**	**[ ]**
4.3	FLUSH	I	**[ ]**	**[ ]**
2.1, 3.1, 3.2	Parameterized derived types, structure constructors, ALLOCATE with SOURCE	T	**[ ]**	**[ ]**
2.7, 2.11, 2.12, 3.5	Type extension incl. polymorphism and SELECT TYPE; fine grained PUBLIC, PRIVATE; PROTECTED	T	**[ ]**	**[ ]**
3.7, 3.8	Pointer assignment und INTENT for pointers	T	**[ ]**	**[ ]**
3.12	ISO_FORTRAN_ENV and intrinsic functions for access to the computing environment	T	**[ ]**	**[ ]**
3.20, 3.21	IEEE arithmetic functions and exception handling	T	**[ ]**	**[ ]**
4.2	Asynchronous I/O, WAIT, ASYNCHRONOUS	T	**[ ]**	**[ ]**
5.2 – 5.7, 2.9	Interoperability with C, BIND(C), ENUM	T	**[ ]**	**[ ]**

2.13.2 Features of the compilers

I 49	The Fortran compiler should provide means to implicitly convert types into others, e.g., to interpret REAL declarations as REAL *8.

[10] See "The New Features of Fortran 2003" by J. Reid, http://www.lrz-muenchen.de/services/software/programmierung/fortran90/new_in_f2k.pdf

	Check here if the requirement can be fulfilled: **[ ]**

I 50:	The Fortran compiler should support integer-type pointers ("Cray Pointers"), a call by value mechanism ("%val"), and means to return the address of a variable ("loc(x)"). Check here if the requirement can be fulfilled: **[ ]**

I 51:	Restricted pointers should be available in C/C++. Check here if the requirement can be fulfilled: **[ ]**

Please explain the various degrees of optimization and the possibilities of the compilers for Fortran, C and C++.

What possibilities exist for **automatic** parallelization and/or vectorization ?

Answers and annotations by the vendor:

[Insert text here]

2.13.3 Third party compilers, libraries and development tools

M 58:	If compilers, libraries, and development tools are delivered by a third party, specify what level of support is available for each delivered package and whether direct access to the support structure of the software provider by LRZ is possible.

M 59:	A suitable licensing including the license fees for compilers, libraries, and development tools must be part of the tender. Please describe the licensing scheme.

M 60:	Specify the vendor's scope of responsibility and warranty for the essential programming environment, particularly compilers, libraries, and performance analysis tools.

Answers and annotations by the vendor:

[Insert text here]

2.13.4 Other parallel compilers and tools

T 33:	In addition, emerging compilers, tools, and libraries are desired, e.g. - Co-Array Fortran - UPC (Unified Parallel C) - Vendor-specific constructs for global data addressing (like SHMEM)

Please describe these compilers and tools, indicating which tools are offered by independent software vendors.

Answers and annotations by the vendor:

[Insert text here]

2.13.5 Message passing

Efficient implementations of message passing libraries must be available for parallelization.

M 61:	MPI (Message Passing Interface) must be available, conforming to the MPI 1.2 standard. Check here if the requirement can be fulfilled: **[ ]**.
M 62:	At least the following features from the MPI-2 standard must be available: - one-sided communication - MPI-IO - extended collective operations - language bindings (Fortran, C, C++). Check here if the requirement can be fulfilled: **[ ]**
M 63:	It must be possible to use the SAN/DAS file systems with high efficiency and performance for MPI-IO. Check here if the requirement can be fulfilled: **[ ]**
I 52:	A thread-safe MPI implementation must be available, i.e., multiple threads may call MPI with no restrictions; MPI_QUERY_THREAD() must return MPI_THREAD_MULTIPLE. Check here if the requirement can be fulfilled: **[ ]**
I 53:	Deadlock detection mechanisms for users and/or system administrators should be provided. Check here if the requirement can be fulfilled: **[ ]**
T 34:	An additional MPI implementation is desired, which, by abandonment of generality and error checking, provides better performance ("light MPI") Check here if the requirement can be fulfilled: **[ ]**
T 35:	It is desired to also use the NAS file system for MPI-IO. Check here if the requirement can be fulfilled: **[ ]**

Please describe your implementation of MPI in detail, particularly special features:

- How is intra-node MPI and inter-node MPI organized?
- To what extent can communication be overlapped with calculation?
- Can MPI processes and/or threads be dynamically created at run time?
- What possibilities does the user have to influence the MPI behavior at run time?
- What profiling interfaces exist for MPI?
- How are signals and aborts handled in MPI processes?

Answers and annotations by the vendor:

[Insert text here]

2.13.6 Other message passing libraries

What other libraries or dialects for message passing are available (e.g., MPICH, LAM, PVM, TCGMSG, ARMCI)?

Answers and annotations by the vendor:

[Insert text here]

2.13.7 SMP parallelism

I 54:	A (POSIX compliant) pthread library should be available. Check here if the requirement can be fulfilled: **[ ]**

Which tools are available for SMP parallel programming, e.g., for binding processes to processors?

Answers and annotations by the vendor:

[Insert text here]

2.13.8 Test aids / debuggers

M 64:	A graphical debugger must be available. Check here if the requirement can be fulfilled: **[ ]**

M 65:	Absolutely necessary for program development are the following test aids: - checking of array bounds and data types during run time. - treatment of IEEE floating-point exceptions. - symbolic dumps and the possibility to analyze core files with a debugger. - possibilities to create trace files from message passing constructs - It should be possible to connect the debugger to an already running (and appropriately compiled) program. Check here if the requirement can be fulfilled: **[ ]**

I 55:	(Etnus) Totalview should be available. Check here if the requirement can be fulfilled: **[ ]**

T 36:	Means for automatic calling user-specified functions at subroutine entry and exit (user hook functions for inserting of user specified performance libraries) are desired. Check here if the requirement can be fulfilled: **[ ]**

Describe the test aids, debuggers, etc. which are available and their features.

Answers and annotations by the vendor:

[Insert text here]

2.13.9 Performance analysis, profiling, and hardware performance monitoring

I 56:	It should be possible to monitor CPU load, use of arithmetic units and vector units (if existing), main memory and cache accesses, data transfers between nodes as well as internal I/O operations. This should be possible from within a user program as well as from a system point of view. Check here if the requirement can be fulfilled: **[ ]**

Which tools are available for analyzing shared memory parallel programs, e.g., OpenMP programs?

Which APIs and tools are available for performance analysis?

Answers and annotations by the vendor:

[Insert text here]

M 66.	Access to the hardware performance counters must be available and be supported.
	Check here if the requirement can be fulfilled: **[ ]**

I 57:	The operating system should provide support for the performance application programming interface (PAPI, see http://icl.cs.utk.edu/papi).
	Check here if the requirement can be fulfilled: **[ ]**

Please describe (preferably in terms of the commonly used PAPI interface) which hardware performance counters are available.

Answers and annotations by the vendor:

[Insert text here]

I 58:	For monitoring message passing programs, Vampir/Vampirtrace (now called Intel Cluster Tools) or a equivalent tool must be available. (The Vampir-GUI itself might run on a workstation or PC).
	Check here if the requirement can be fulfilled: **[ ]**

T 37:	The collection of data should require the least possible amount of resources and be non-conflicting (user programs should not influence each other in measurements, an evaluation by the user should not disturb or hinder data collection by the system administrator).

Please fill in the following table indicating what level of support is available for hardware performance counters (HWPC) and other performance data.

HWPC are available for every node and can be accessed by a system administrator without intervention of a user.	**[ ]**
HWPC are available for every CPU and can be accessed by a system administrator without intervention of a user.	**[ ]**
HWPC are available for every job. They can be accessed by a user or a system administrator. Detailed information can be accessed after job completion.	**[ ]**
HWPC are available for every user application or process and can be accessed by a system administrator without the intervention of a user.	**[ ]**
Means for the automatic profiling of applications with HWPC on at least subroutine level are available.	**[ ]**
Accessing HWPC by a system administrator does not interfere with user-induced profiling activities.	**[ ]**

Answers and annotations by the vendor:

[Insert text here]

2.13.10 Parallel programming environment

T 38:	An integrated programming environment that allows to develop, start, debug and optimize parallel programs and to visualize the performance data is desired. Check here if the requirement can be fulfilled: **[ ]**

If one is available, please describe its functions.

Answers and annotations by the vendor:

[Insert text here]

2.13.11 Third Party Software

Please indicate the present implementation state of important products for the following areas.

- Computational chemistry, biology, and chemical engineering:
  AMBER, BLAST, CHARMm, CPMD, DISCOVER, GAMESS, GAUSSIAN, MOLPRO, MOPAC, NWChem, VASP
- Computational fluid dynamics:
  CFX, FIRE, FLOW-3D, FLUENT, LS_Dyna, PAMFLOW, STAR-HPC
- Structural mechanics:
  ABAQUS, ANSYS, DYNA3D, MARC, MSC/NASTRAN, PAM-CRASH, RADIOSS
- Mathematics and Statistics
  BLAS, BLACS, LAPACK, ScaLAPACK
  IMSL, NAG, GNU Scientific Library (GSL),
  ATLAS
- Graphics and Visualization:
  AVS, VTK, Vis5D, SCIRun/BioPSE,Data Handling:
  NetCDF, HDF5, PACT
- Communication and Metacomputing:
  Global Arrays, PACX, MPICH-G2, MpCCI, VISIT
- Grid Software:
  Globus, UNICORE

Answers and annotations by the vendor:

[Insert text here]

M 67:	Highly efficient parallel numeric libraries and other important application programs and libraries must be available. In particular **64-bit versions** of the following products must be provided: - a scientific library with highly optimized routines for the offered system (e.g., with FFT, random number generator, linear algebra etc.). - besides the serial versions, SMP-parallel versions of BLAS and LAPACK must be provided - ScaLAPACK, BLACS - PETsc - FFTW - NAG Fortran Libraries (serial, parallel, and SMP-parallel) - Global Arrays Check here if these requirements can be fulfilled: **[ ]**

M 68:	Parallel versions of CPMD and NWChem must be available
	Check here if the requirement can be fulfilled: **[ ]**

M 69:	UNICORE and Globus must be available.
	Check here if the requirement can be fulfilled: **[ ]**

I 59:	Check the appropriate boxes, if efficient, parallel implementations of the following libraries are available	
	PLAPACK	**[ ]**
	MpCCI	**[ ]**
	PACX-MPI	**[ ]**
	MPICH-G2	**[ ]**
	HDF5	**[ ]**
	NetCDF	**[ ]**

T 39:	A shared memory parallel version of GAUSSIAN is desired.
	Check here if the requirement can be fulfilled: **[ ]**

2.13.12 Shells and scripting languages

M 70:	At least the following shells must be supported (in 64-bit mode):
	sh/ksh, bash, csh, tcsh.
	Check here if the requirement can be fulfilled: **[ ]**

M 71:	perl must be provided.
	Check here if the requirement can be fulfilled: **[ ]**

I 60:	Python and Tk/Tcl should be available.
	Check here if the requirement can be fulfilled: **[ ]**

T 40:	A Java runtime environment is desired.
	Check here if the requirement can be fulfilled: **[ ]**

2.13.13 GNU development tools

I 61:	A port of commonly used development tools from the GNU tool chain (e.g., gmake, gawk, autoconf/automake, cvs, bison) should be available.
	Check here if the requirement can be fulfilled: **[ ]**

Please describe which tools are supported.

Answers and annotations by the vendor:

[Insert text here]

2.14 Training, user support and technical support

2.14.1 Documentation

The description (in English or German) of the system and the software products should be available on the following levels:

- introductory descriptions for end users
- online information (in HTML and/or PS/PDF),
- explanations about error messages (system, compiler, run time system, etc.) should be provided online
- complete user instructions for all products and the operating system.

What are the licensing regulations for online-supply of documentation (e.g., access limitations on the basis of IP addresses, password protection)?

M 72:	Manuals and/or online documentation describing the system and its environment must be provided. Check here if the requirement can be fulfilled:	**[ ]**

A list of manuals must be submitted

Answers and annotations by the vendor:

[Insert text here]

2.14.2 Introduction to the system

M 73:	Before the system is put into operation, an introduction for selected users and LRZ staff must take place, informing about programming (including compilers and libraries) and system software (computer configuration and handling). Check here if the requirement can be fulfilled:	**[ ]**

2.14.3 User support

A supercomputer requires staff-intensive user support. LRZ depends on the manufacturer's assistance for user support. Hence:

M 74:	During the operation of the computer at LRZ, the vendor must provide one software engineer with deep knowledge about the operation of the system, who assists with optimization related problems of users. Check here if the requirements can be fulfilled:	**[ ]**

T 41	Please describe additional possibilities you could offer to assist LRZ's users in getting optimal performance.

2.14.4 System-related support

M 75:	One engineer must be available for the whole time of the operation of the system, who will be in charge of system-related support. Check here if the requirements can be fulfilled: **[ ]**

M 76	During the first half year of the HLRB II Phase 1 as well as Phase 2 operation, the vendor must provide **additional on-site** personnel which, together with LRZ staff, is in charge of configuring and putting the system into operation. This personnel must have excellent knowledge of the system and must maintain the necessary contacts to the vendor. Check here if the requirements can be fulfilled: **[ ]**

T 42	Please describe additional possibilities you could offer to LRZ with system administration and tuning.

Office space for all vendor personnel will be provided by LRZ.

Answers and annotations by the vendor:

[Insert text here]

2.15 Maintenance

2.15.1 Hardware maintenance

M 77	A maintenance contract fulfilling the following conditions must be offered: During peak operation times (on normal working days, from 8 am till 6 pm), response to a hardware error must happen within four hours. Check here if the requirements can be fulfilled: **[ ]**

I 62	Detection of hardware errors should be carried out automatically and be automatically reported to the vendor. Check here if the requirements can be fulfilled: **[ ]**

Further details about guaranteed repair times and possible delivery times for spare parts will be regulated in the maintenance contract.

2.15.2 Software maintenance

M 78	A maintenance contract corresponding to the following conditions must be offered: During peak operation times (on normal working days, from 8 am till 6 pm), response to a software error must happen within four hours. Check here if the requirements can be fulfilled: **[ ]**

2.15.3 Total cost of maintenance

M 79:	The total cost for hardware and software maintenance for a 5 year term of operation must be specified in the following table:

Answers and annotations by the vendor:

[Insert text here]

Total cost for HW maintenance:	_____ €
Total cost for SW maintenance:	_____ €

2.16 Collaboration with the vendor

T 43: A close collaboration with the vendor is desired.

If offered, describe the scope and extent of such a collaboration.

Answers and annotations by the vendor:

[Insert text here]

3 Benchmarking procedure for the HLRB II

3.1 Overview of the benchmarking procedure

The LRZ benchmark suite for the procurement of the HLRB II consists of four major parts:

Interconnect-related benchmarks which include:

- Pallas MPI-1 Benchmark for measuring the interconnect bandwidth and latency
- Pallas MPI-2 Benchmark for measuring the bandwidth and latency for one-sided communication

Storage subsystem benchmarks

- IObench: a program for measuring the aggregate I/O throughput
- MPI-IO Benchmark: for measuring the MPI-IO performance
- Metadata: for measuring file transaction rates

Architectural ("Kernel") benchmarks which include five small application programs:

- RINF1: for testing the floating point and integer performance of loop kernels.
  The benchmark part "Contiguous Triads" provides a rough estimate of the performance which can be expected from typical engineering applications; the performance numbers are essentially dictated by the memory bandwidth of the system node, but large caches can have a positive performance impact.
- SipSolver: strongly implicit procedure solver for typical CFD-Kernels.
  Vendor specific solutions are permitted.
- FFT: for testing the performance of Fourier transforms.
  Vendor specific library solutions are encouraged.
- ZHEEVD: for computing the solutions of complex hermitian eigenvalue problems.
  Vendor specific BLAS and LAPACK solutions are encouraged.
- DMRG: density matrix renormalization group algorithm.
  C++ Code with OpenMP directives.
- LASER: Helmholtz eigenvalue problem arising from a laser simulation.
  C++ code, using expression templates
- LINPACK.
  A well known standard HPC benchmark

Application Benchmarks which include typical programs from potential LRZ users:

- BEST: a Lattice-Boltzmann code for fluid dynamics (Chair for Fluid Mechanics, University Erlangen-Nuremberg, Prof. Durst)
- BQCD: quantum chromodynamics code of the QCD-Collaboration (Konrad-Zuse-Zentrum für Informationstechnik Berlin, H. Stüben)
- Cactus: astrophysics (Albert-Einstein-Institute Potsdam)
- HEPFP: spectral properties of atomic Rydberg states in intense electromagnetic fields (Max-Planck-Institute for the Physics of Complex Systems)
- MGLET: a multigrid large eddy simulation code (Chair for Fluid Dynamics, Technical University Munich, Prof. Friedrich)
- NWChem: a scalable quantum chemistry code (Pacific Northwest National Laboratory)

3.1.1 Confidentiality

Under no circumstances shall any information related to the design, involved algorithms, or source code of the benchmarks be disclosed to a third party without written consent of LRZ and the authors of the particular benchmark.

3.1.2 Code modifications

Any modifications which do not change the intention and spirit of the underlying algorithms or lower the precision of the calculations are permitted. The changes must be at least as robust as the baseline algorithm. Typical examples of allowable modifications include:

- Insertion of compiler directives
- Loop unrolling or fusion
- Blocking
- Changes of data layout or of the alignment of data
- Calls to library subroutines
- Inlining
- Reversing outer and inner loops
- Use of threads (implicit or explicit)
- Pattern matching techniques for replacing original code with calls to libraries
- Replacement of message passing constructs by shared memory constructs or by one-sided communication constructs.

Any change of the source code must be fully disclosed and will be discussed with the authors of the codes. A short rationale for any change must be provided.

Compilation or linkage flags which are supported and documented by the vendor are permitted.

Linking to optimized versions of vendor libraries is permitted and encouraged. The usage of all libraries must be disclosed together with the submission of the results.

Source code directives which are supported and documented by the vendors' compiler may be used in the benchmark codes. Language extensions instead of directives may also be used.

3.1.3 Hints for optimization

On systems with non-flat shared memory it may be necessary to perform initialization of data with the same layout than the actual computation, particularly in systems with "first touch" policies.

It may be also necessary to revert the Fortran90 notation to Fortran77 to obtain the optimal placement of data on NUMA-systems.

3.1.4 Limitations of optimization

We take it for granted that the vendor understands the intention of the loop kernels (especially those of the low-level benchmarks) and undertakes no action to circumvent the intended measurements.

The following optimizations are **not** permitted:

- **Code to circumvent the actual computation:** Any modification of the code to circumvent the actual computation is not permitted, e.g., it is not permitted to use Strassen's algorithm for DGEMM matrix multiplication. This prohibition should of course not cover standard optimization techniques, whether by the compiler or carried out manually, which pull redundant code out of the loops and precompute it.
- **Eliminate code or bypass code** which is not covered in the actual benchmark case but may be covered in other benchmark cases.
- **Optimized assembly modules:** Substituting any part of the code by optimized assembly modules or modification of compiler generated assembly code or executables is discouraged and should be disclosed when reporting the results. In this case the performance of the non-substituted code must also be disclosed.
- **Pre-supposing the knowledge of the solution:** Any modification of the code or input data set which makes use of known properties of the solution or of the verification tests is not permitted.

3.1.5 Implementation of basic data types

If not stated otherwise the calculations should be carried out with data types defined as follows:

- REAL, REAL*4 means at least 32 bit wide data type
- REAL*8, DOUBLE PRECISION means at least 64 bit wide data type
- INTEGER means at least 32 bit wide data type
- INTEGER*8 means at least 64 bit wide data type.

The MGLET application benchmark is an exception from this rule. Please see the description of this benchmark for details.

3.1.6 Conversion of timing to performance values

For most of the programs timing values are converted to floating point operations per second by dividing a predefined operation count by the execution time (wall clock time). However, in some cases only parts of the program are measured, e.g., omitting the initial phase or the final phase.

It is not allowed to change the conversion factors, **even if they may appear wrong or inappropriate**.

For any given benchmark program we consider the converted values as just another measurement unit of execution time, which serves to rank the results against the best performance obtained among vendors.

3.1.7 Rules for systems which are not available for benchmarking

The description of each benchmark program specifies which results need to be delivered. In case that a benchmark program cannot be run on the proposed system, either because there is no system of the offered size available or because the proposed system hardware or software can not yet be benchmarked at all, predictions have to be made by the vendor.

These predictions are considered as committed minimal performance results. They should be based on measurements done on a roughly comparable system which is presently available and is similar in architecture to the one offered. Results obtained on smaller systems need to be carefully scaled to the predicted values for the proposed system size or to the required benchmark size. Together with the predictions, the results of actual measurements must also be disclosed in order to enable LRZ to perform a qualitative evaluation of the characteristics of the offered system.

All mandatory commitments by the vendor are marked red, like the section here. Committed values have to be demonstrated by the vendor during the acceptance procedure for the system. Failure to reproduce the committed performance requires that appropriate countermeasures need to be taken by the vendor.

Additional information in the commitment boxes not marked red is only used for qualitative evaluation. It also indicates how the committed values were computed.

The reasoning behind all estimates and predictions should be explained. A rationale for the assumptions underlying the predictions should be given.

3.1.8 Failure to reach the required minimum performance values

For most of the benchmarks minimum requirements are specified. If the vendor fails to reach a minimum performance requirement, the corresponding benchmark will be counted with zero performance in the ranking procedure.

Nevertheless the vendor has to cite his best efforts to reach the required minimum values.

3.1.9 Commitments for installation Phase 1 and 2

As specified elsewhere, this request for proposal is for a supercomputer to be installed in two phases: Phase 1 in late 2005 and Phase 2 in the year 2007.

The vendor must disclose and commit detailed performance results for Phase 1.

During the acceptance test for Phase 1 the performances of the individual benchmarks and the overall performance have to be demonstrated. The performance of individual benchmarks may fail to reach the committed values **by a margin of 5%** but the commitments for the overall computing performance for Phase 1, as defined in 5.1, must be reached or exceeded.

During the acceptance test for Phase 2 the performances of the individual benchmarks and the overall performance have to be demonstrated. The performance of individual benchmarks may fail to reach the committed values **by a margin of 10%** but the commitments for the overall computing performance for Phase 2, as defined in 5.1, must be reached or exceeded.

3.1.10 Disclosure by the vendor and submission of results to LRZ

The following paragraphs detail the information constituting a full disclosure:

- description of the hardware and software configuration
- system tuning options:
- I/O configuration:
- benchmark code tuning options:
- source code including all modifications and rationale for any change.
- Result files, as produced by the benchmark programs

All required results and the source code of the programs should be supplied to LRZ on a CDROM.

3.1.11 Disclosure of the optimizations to the authors of the benchmarks

LRZ will disclose the optimizations for the application benchmarks to the respective authors of the particular program and will discuss the modifications with them.

3.1.12 Delivery of the benchmark sources by LRZ

The final version of the benchmark source codes for the HLRB II procurement is delivered by LRZ on a CDROM. All preliminary versions of the benchmarks and the accompanying documentation are then considered obsolete.

3.2 Aggregate computing performance

The term *aggregate computing performance* refers to a method to calculate the performance of a fully loaded system for any given benchmark.

First, the vendor has to decide for each of the benchmark programs, how many CPUs must be used to achieve or exceed a required minimum computing performance (typically given in GFlop/s) of the program. The performance numbers need to be extracted from the result files of the programs. For each program, the vendor can decide to use a higher number of CPUs than actually necessary to achieve the required minimum performance. In particular, this can make sense if an SMP node otherwise would not be fully utilized.

Subsequently, the vendor has to measure (or stipulate) which is (or will be) the **aggregate performance** of **all compute processors** of the offered supercomputer when simultaneously running as many identical copies of the program as possible with at least the required minimum performance for every individual run.

In simplified terms (neglecting saturation, shared memory and interconnect effects, etc.), the method of performance determination can be described as follows:

> *The vendor ascertains the average computing performance of a program per processor (ensuring that the required minimum performance for the entire program is achieved) and then multiplies this performance value with the number of compute processors.*

This aggregate computing performance should be maximized. For this reason, the vendor will endeavor to use all compute processors of the offered supercomputer.

However, the vendor must take into account that under certain circumstances (e.g., because of bottlenecks of central memory access or of the interconnect) the aggregate performance for running **n** identical copies of a program is not always simply **n** times the performance of a single program run.

Typically, the vendor endeavors to run the programs only with as many processors as necessary to achieve the required minimum performance, due to the fact that efficiency decreases with increasing number of processors. It may happen that a number of "M" processors of a total of "N" offered compute processors remain unused or do not allow to achieve the minimum performance required for the program. In this case, the vendor is allowed (as a relaxation of the above rule to run only identical copies) to take the following action:

- fill the system asymmetrically with program copies, as long as each single copy achieves the required minimum performance in an individual run (this must be proved separately). In this case the total aggregate performance is the sum of performances of all program copies,

or

- run a version of the program with a smaller problem size on the remaining processors as a dummy program to produce some workload,

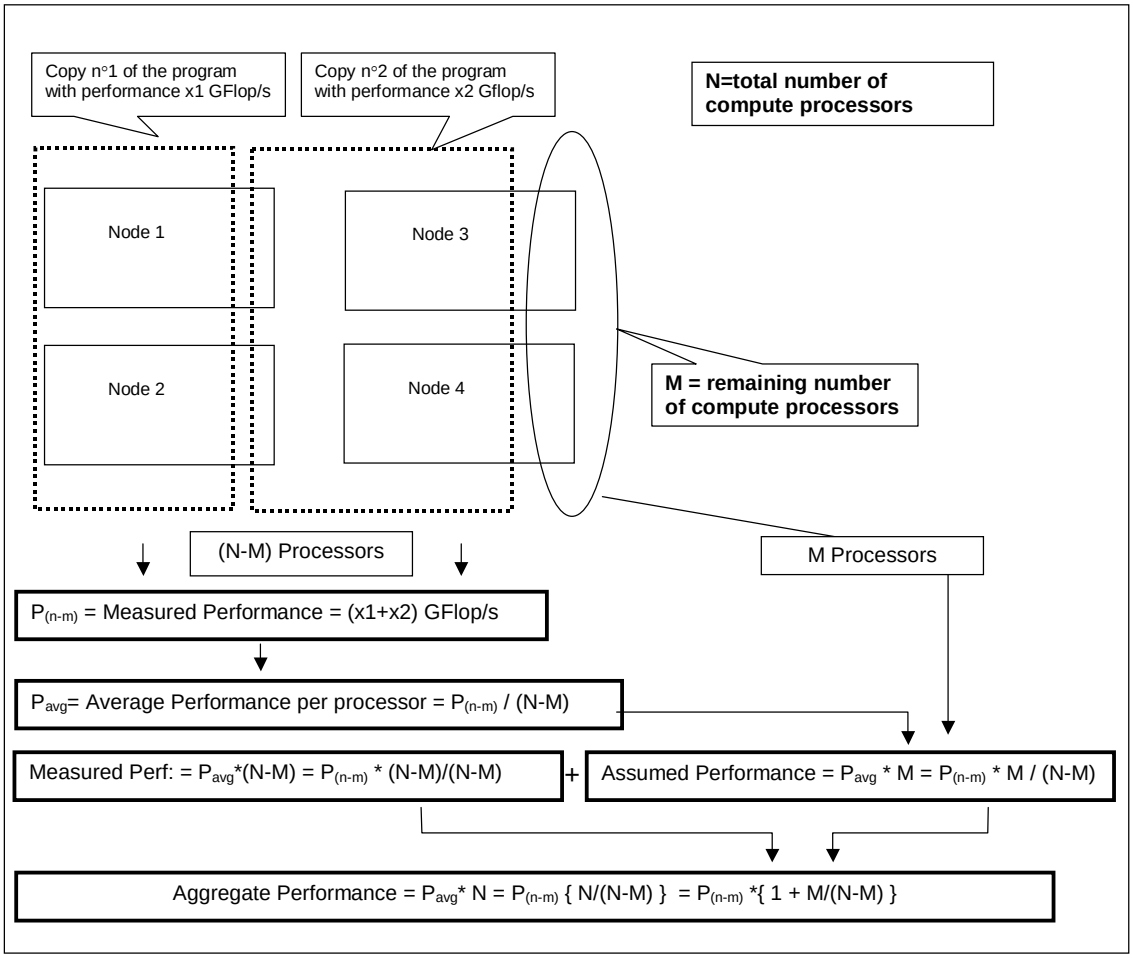
or

- start copies of the benchmark program low_level/rinf1 as a dummy program to produce some artificial workload.

The dummy programs are only used to model memory or interconnect bottlenecks. In the two latter cases, the vendor determines the aggregate computing performance of the whole system by **linear extrapolation** of the performance of the actually measured programs (i.e., the sum of the performances of n copies) to the full-sized system. For this purpose he multiplies the actually measured performance or sum of performances, P_m, with the following factor:

$$\frac{N}{(N-M)} = \left(1 + \frac{M}{(N-M)}\right)$$

The method of ascertaining the aggregate computing performance is also illustrated by the following figure.



Scheme for the calculation of Aggregate Computing Performance

Since the effect of memory bottlenecks in shared memory systems and congestion of the internal network has to be taken into account, the programs (and, if present, the dummy programs filling the remaining M processors) must run simultaneously. This implies that the programs are started at the same time. The start time of the earliest program is considered the start time for all programs; the time of termination for each program is its individual time of termination. Then, the run time of a program is calculated as the difference of starting time and termination time. The run time of the longest-running program must not exceed 1.2 times the run time of the shortest-running program. If there remain any justified doubts that the programs run simultaneously, the vendor must prove the simultaneous execution by insertion of further time stamps in the program code.

3.3 Infrastructure for the benchmark suite

3.3.1 Setting up the source tree

The benchmark suite is packaged as a gzip-compressed UNIX archive “bench.tar.gz”. To unpack the archive using the standard UNIX “tar” command, please type

```
gunzip bench.tar.gz
tar –xf bench.tar
```

The recommended procedure is to use GNU tar to unpack the archive:

```
gtar –xzf bench.tar.gz
```

On Linux-based systems, GNU tar will usually be available as the standard “tar” command. After unpacking the archive, please proceed by typing

```
cd bench
. ./setup.sh
```

while working under a Bourne-derived shell (e. g., bash, ksh). At this point you might find that your system is not supported, in which case you need to

(1) make suitable changes to `setup.sh`: If your operating environment is not supported, add a switch branch for your value of $OPERATING_SYSTEM. If your processor type is not supported, add suitable settings for processor type ($ARCH) and SMP size ($SYS_NUM_PROCS).

(2) change to the `etc` subdirectory and make a copy of a `make.inc*` (possibly the one with the nearest relationship to your architecture):

```
cp make.inc.whatever make.inc.$OPERATING_SYSTEM.$ARCH
```

This include file will be generically used for many of the benchmark programs and also serves as a reference for looking up compilation settings etc. Make sure the appropriate environment variables are set correctly for this initial setup! Then, edit this new file to adjust for suitable settings appropriate for the system you intend to perform the testing on.

After adding support for your system, it is probably necessary to rerun `setup.sh`. Note that the benchmark root directory name is stored in the environment variable $BENCH by this script. Please also consult the file $BENCH/README for further details on setting up the benchmark suite.

3.3.2 Low-level library used by the benchmark programs

The directory `$BENCH/src/aux` contains routines used by more than one benchmark program, e.g., timing routines for wall clock time and CPU time. Modify the files `time.c, time.body.c,` and `time.body1.c` to include the machine-specific timing routines which should measure the wall-clock and CPU time in seconds. To build the library, type

```
cd $BENCH/src/aux
gmake
```

3.3.3 Building an executable

Each benchmark program is contained in a separate subdirectory which also contains its own Makefile.

After making machine-specific code modifications in the subdirectories, you can compile the benchmark programs. To compile a single benchmark program, use:

```
cd $BENCH/src/{category}/{bm}
gmake
```

where {category} stands for one of the categories (e.g., Apps) and {bm} stands for the specific benchmark (e.g., MGLET). Note that some benchmark Makefiles refer to the common include-file

```
$BENCH/$MAKEINC
```

contained in `$BENCH/etc` as mentioned above. Makefiles, include files and scripts are provided for convenience and reference only. They are not part of the benchmark. The vendor may replace them by his own optimized settings and/or scripts.

3.3.4 Running a benchmark executable

A sample script `$BENCH/bin/run` is provided which can be used to start up the benchmark programs. In addition to the internal time measurement within the benchmark programs this script must perform an external time measurement.

The script takes at least three input parameters beyond the executable to be run:

- -n: the (total) number of MPI processes
- -N: the number of nodes
- -t: the number of threads

Modifications: Modify the script according to your needs, e.g., insert calls to `mpiexec` to start the executables. Using other calls than the `time` command which give additional information about the performance of the system (e.g., Flop-counters, hardware performance monitors/counters) is appreciated, but these commands must at least report the elapsed (real) time, user time, and system time.

4 Benchmarks

4.1 Interconnect-related benchmarks

These benchmarks provide a set of MPI kernels. They are based on the Pallas MPI Benchmark suite (PMB) with slight modifications by LRZ. In particular, the definition of throughput for the pingping communication pattern has been changed. Also, reading in a configuration file `"assign_nodes"` in subroutine `readconfig.f` was enabled. At the beginning of the benchmark run, a new communicator MPI_COMM_WORLD_X is created, which reflects an optionally changed mapping of process ranks to processors. All subsequent communicators are derived from MPI_COMM_WORLD_X.

In the multi-process group version of the benchmark, disjoint groups of 2, 4, 8, etc. processes will be formed, which will all simultaneously run the benchmark routines.

A detailed description of the benchmark can be found at: http://www.pallas.com/e/products/pmb/d-reply.htm.

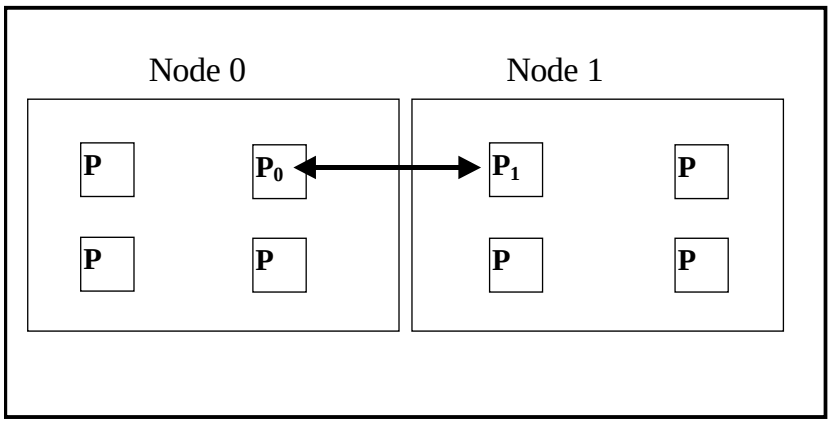
Please note that the measure of **system balance** is not **interconnect bandwidth** versus **peak performance**, but **interconnect bandwidth** versus the **weighted performance of the RINF1** double precision vector triad (Case 2) as described in section 4.3.1. The latter number is strongly related to the memory bandwidth of a node and is a measure for the expected sustained performance.

4.1.1 MPI-1 Benchmark: Link bandwidth of a node, one MPI task per node

Purpose:	Measure the bandwidth for programs which run like MPI_THREAD_SINGLE, MPI_THREAD_FUNNELED, or MPI_THREAD_SERIALIZED
Source:	src/low_level/mpi
Executable:	PMB-MPI1
Compile	gmake PMB-MPI1
Procedure:	Run PMB-MPI1 between two nodes. Run **exactly one** MPI task on each node. Let the other CPUs of the node idle.

Command line:	`$BENCH/bin/run -n 2 -N 2 -t 1 \`
	`  PMB-MPI1 pingpong pingping sendrecv exchange\`
	`  | tee > mpi.single.out`

Example:	Two nodes, each having 4 processors. Only one processor of each node takes part in the communication



Results: Deliver `mpi.single.out`

Report the bandwidth value for the message size of 4 MByte (=4194304 Bytes) for the four cases specified above. The highest of these four values is taken and multiplied with the number of nodes in Phase 1 divided by 2.

Commitment for Phase 1:

Bandwidth for pingpong = _____ GByte/s
Bandwidth for pingping = _____ GByte/s
Bandwidth for exchange = _____ GByte/s
Bandwidth for sendrecv = _____ GByte/s
(these four values are only for reference and qualitative evaluation)

MPI link bandwidth (one MPI task per node) = _____ GByte/s
(Max of above)

This MPI link bandwidth is put into relation with the compute performance of an SMP MPI task. We expect approximately 8 threads per MPI task, but some vendors may have only 4 very powerful CPUs per node. Thus the minimum of these two values has to be taken and must be multiplied with the per CPU performance for the vector triad as an approximation for the sustained performance.

Interconnect balance for MPI_THREAD_SERIAL =
link bandwidth (Byte/s) for one MPI task
(as defined by the MPI benchmark above)
divided by
{ Minimum (number of CPUs per node, 8) times
maximum per-CPU performance "T1" for the double precision vector triad
(as defined by Case 2 of the RINF1 benchmark in section 4.3.1) } =
_____ **(Byte/s) / (Flop/s)**

4.1.2 MPI-1 Benchmark: Saturated link bandwidth and interconnect balance of a node

Purpose: Measure the aggregate bandwidth for a program which runs like MPI_THREAD_MULTIPLE. Especially, the saturation bandwidth of a node is measured.

Source: `src/low_level/mpi`

Executable: PMB-MPI1 (same as in 4.1.1)

Procedure: Run PMB-MPI1 between Node 0 and a sufficient number of other nodes to saturate the links of Node 0.

All processes with even process IDs must run on Node 0, which can be accomplished by using a hostfile or by using the subroutine `readconfig.f` and/or the configuration file `assign_nodes`.

The number of processes taking part in this benchmark run is denoted NPROCS; this is an even number which is chosen for optimal performance numbers by the vendor.

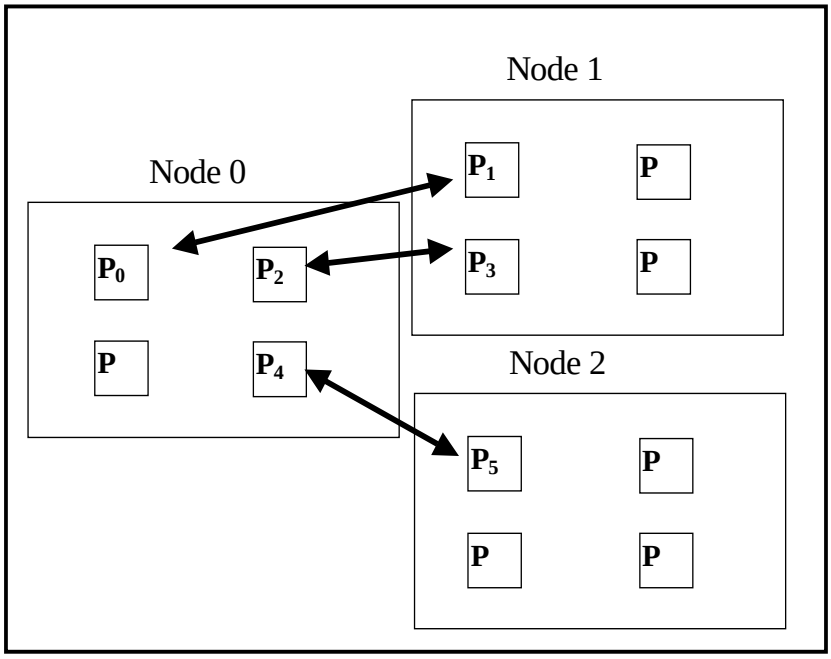
Since we run the "multi" version of the benchmark, disjoint groups of processes are formed. Therefore, the reported bandwidth must be scaled with the number of processor pairs; this is half the number of MPI processes taking part in the benchmark.

Aggregate saturated bandwidth of a node =
(reported bandwidth) times (NPROCS/2)

Command line:

```
$BENCH/bin/run -n $NPROCS -N <nodes> -t 1 \
  PMB-MPI1 pingpong pingping sendrecv exchange \
    -multi 0    | tee mpi.multiple.out
```

Example: Three processors on Node 0 are communicating with two processors on Node 1 and one processor on Node 2. Thus, the reported bandwidth values have to be scaled by a factor of 3.



.

Results: Deliver `mpi.multiple.out`

Take the bandwidth value for the case where "NPROCS/2 groups of 2 processors" are formed, i.e., MPI Communication Group 0 consists of the processes 0 and 1.

Report the bandwidth value for the **message size of 4 MByte** (=4194304 Bytes) for the four cases specified above and multiply the value by half the number of processors taking part in the communication. The highest of these four values must meet the commitment by the vendor for the **saturation bandwidth of one node**.

Commitments and requirements:

Commitment:

Reported Bandwidth for pingpong * (NPROCS/2) =	_____ GByte/s
Reported Bandwidth for pingping * (NPROCS/2) =	_____ GByte/s
Reported Bandwidth for exchange * (NPROCS/2) =	_____ GByte/s
Reported Bandwidth for sendrecv * (NPROCS/2) =	_____ GByte/s

(these four values are only for reference and qualitative evaluation)

Aggregate MPI link bandwidth of one node
(Max. of above) _____ GByte/s

Interconnect balance of a node =
aggregate MPI link bandwidth (Byte/s) of one node
(as defined by the MPI benchmark above)
divided by
the weighted performance of the vector triad (Flop/s) of one node
(as defined by Case 2 of the RINF1 benchmark in section 4.3.1)
= _____ **(Byte/s) / (Flop/s)**

Reminder: The interconnect balance should exceed 0.4 (Byte/s) / (Flop/s).
This was already set forth as a requirement in 2.3.2).

4.1.3 MPI-1 Benchmark: Bisection bandwidth and latency within a Phase

Purpose: Measure the bisection bandwidth within the nodes of one phase.

The bisectional bandwidth is the minimum bandwidth of all possible bisection bandwidths (worst case bisectional configuration within one phase).

Source: `src/low_level/mpi`

Executable: PMB-MPI1

Procedure: Divide all nodes of one installation phase into two equally sized sets, called “left” and “right” in the following. Use as many processors on a node as you need to reach the maximum aggregate bandwidth of the internal network, but **at least one processor of each node** must take part in the communication.

Run the benchmark with pairs of processors, where the two processors of a pair are on different sides of the configuration.

All even numbered MPI Processes must run in the "left" set, all odd numbered MPI processes must run in the "right" set of nodes. This can easily be accomplished by using a hostfile or by using the subroutine readconfig.f and/or the configuration file `assign_nodes`.

The number of processes taking part in this run is denoted NPROCS.

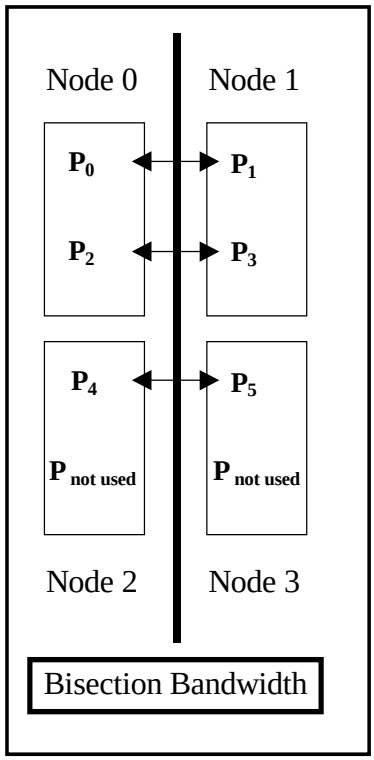
The reported bandwidth must be scaled with the number of processor pairs which is half the number of MPI processes taking part in the benchmark.

aggregate bisection-bandwidth of a phase =
(reported bandwidth) times (NPROCS/2)

Command line:

```
$BENCH/bin/run -n $NPROCS -N $NODES -t 1 \
  PMB-MPI1 pingpong pingping sendrecv exchange \
    -multi 0  | tee mpi.bisection.out
```

Example: 4 nodes with 6 MPI processes take part in this example. The reported bandwidth of the benchmark has to be scaled by a factor of 3.



Results: Deliver `mpi.bisection.out`

Take the bandwidth value for the case where "NPROCS/2 groups of 2 processors" are formed, i.e., MPI Communication Group 0 consists of the processes 0 and 1.

The message size of 4 MByte (=4194304 Bytes) is used for the test cases.

The highest bandwidth of the four cases specified above must meet the commitment by the vendor for the bisection bandwidth per node and is multiplied with the number of processes taking part in the communication divided by 2.

Take the timings for the case where "NPROCS/2 groups of 2 processors" are formed. The execution time for a message size of 1 Byte is used as definition of the MPI latency.

The lowest timing for the four cases specified above must meet the commitment by the vendor for the latency.

Commitments and requirements:

Bandwidth Commitment:

Bandwidth for pingpong * (NPROCS/2) = _____ GByte/s
Bandwidth for pingping * (NPROCS/2) = _____ GByte/s
Bandwidth for exchange * (NPROCS/2) = _____ GByte/s
Bandwidth for sendrecv * (NPROCS/2) = _____ GByte/s

(these four values are only for reference and qualitative evaluation)

Aggregate bisectional bandwidth of Phase 1 (Max. of above) = _____ GByte/s

Latency Commitment for Phase 1:

Time for pingpong with 1 Byte and NPROCS/2 groups =	_____ µs
Time for pingping with 1 Byte and NPROCS/2 groups =	_____ µs
Time for exchange with 1 Byte and NPROCS/2 groups =	_____ µs
Time for sendrecv with 1 Byte and NPROCS/2 groups =	_____ µs

(these four values are only for reference and qualitative evaluation)

MPI latency of Phase 1
(Min. of above) = _____ µs

Reminder: The MPI latency, as defined here, must not exceed 8 µs within Phase 1 as well as within Phase 2.

This was already set forth as a requirement in section 2.3.2.

4.1.4 MPI-1 Benchmark: Bisection bandwidth and latency between Phase 1 and Phase 2

Purpose: Measure the bisection bandwidth between Phase 1 and Phase 2. The bisectional bandwidth is the minimum bandwidth of all possible bisection bandwidths (worst case bisectional configuration within one phase).

Source: `src/low_level/mpi`

Executable: PMB-MPI1

Command:

```
$BENCH/bin/run -n $NPROCS -N $NODES -t 1 \
  PMB-MPI1 pingpong pingping sendrecv exchange \
       -multi 0  |  tee mpi.bisection.ph1ph2.out
```

Procedure: Create pairs of processors between Phase 1 and Phase 2.

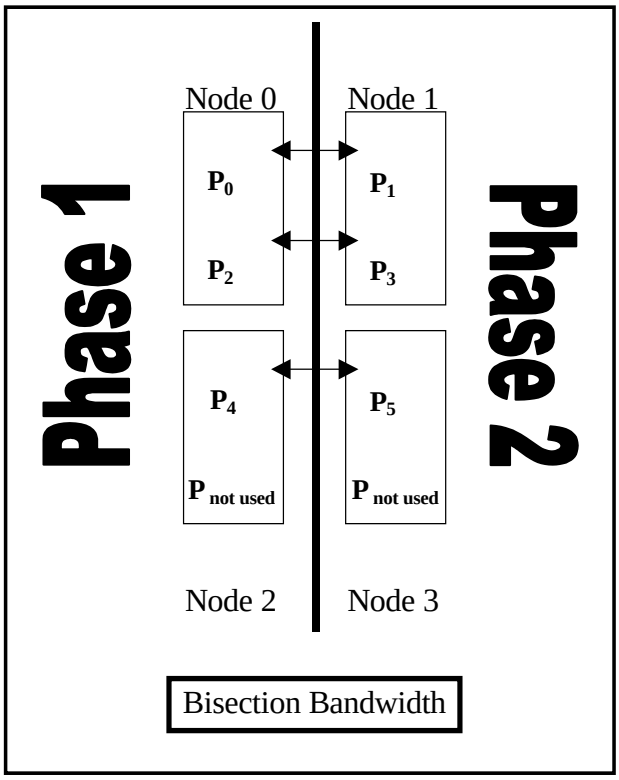
Use as many processors as you need to reach the maximum aggregate bandwidth of the network, but **at least one processor of each node** must take part in the communication.

All even numbered MPI processes must run on the "Phase 1" side, all odd numbered MPI processes must run on the "Phase 2" side of the configuration. This can easily be accomplished by using a hostfile or by using the subroutine readconfig.f and/or the configuration file `assign_nodes`. The number of processes taking part is denoted NPROCS.

The reported bandwidth must be scaled with the number of processor pairs which is half the number of MPI processes taking part in the benchmark.

aggregate bisection-bandwidth of a Phase =
(reported bandwidth) times (NPROCS/2)

Example: 4 nodes with 6 MPI processes take part in this example



Results: Deliver `mpi.bisection.ph1ph2.out`

Take the bandwidth value for the case where "NPROCS/2 groups of 2 processors" are formed, i.e., MPI Communication Group 0 consists of the processes 0 and 1.

The message size of 4 MByte (=4194304 Bytes) is used for the test cases. The highest bandwidth of the four cases specified above is taken as the commitment by the vendor for the bandwidth and is multiplied with the number of processes taking part in the communication divided by 2.

Take the timings for the case where "NPROCS/2 groups of 2 processors" are formed. The execution time for a message size of 1 Byte is used as definition of the MPI latency. The lowest timing for the four cases specified above is taken as the commitment by the vendor for the latency.

Commitments and requirements:

Bandwidth Commitment for Interconnect between phase 1 and phase 2:

Bandwidth for pingpong * (NPROCS/2) = _____ GByte/s
Bandwidth for pingping * (NPROCS/2) = _____ GByte/s
Bandwidth for exchange * (NPROCS/2) = _____ GByte/s
Bandwidth for sendrecv * (NPROCS/2) = _____ GByte/s

(these four values are only for reference and qualitative evaluation)

Aggregate bisectional bandwidth between phase 1 and phase 2 (Max. of above) = _____ GByte/s

Latency Commitment for Interconnect between phase 1 and phase 2:

Time for pingpong with 1 Byte and NPROCS/2 groups = _____ µs
Time for pingping with 1 Byte and NPROCS/2 groups = _____ µs
Time for exchange with 1 Byte and NPROCS/2 groups = _____ µs
Time for sendrecv with 1 Byte and NPROCS/2 groups = _____ µs

(these four values are only for reference and qualitative evaluation)

Latency between Phase 1 and Phase 2 (Min. of above) = _____ µs

Reminder: The MPI bidirectional bisection bandwidth between Phase 1 and Phase 2, as defined here, must exceed 80 GByte/s (40 GByte/s in each direction)

This was already set forth as a requirement in 2.3.3).

Reminder: The MPI latency (as defined here) across the phase boundaries between two arbitrary nodes should not exceed 10 µs.

This was already set forth as a requirement in section 2.3.3.

4.1.5 MPI-2 Benchmark: Bisection bandwidth for one-sided communication within a Phase

Purpose: Measure the bisection bandwidth for one-sided communication (put/get) within a phase. The bisectional bandwidth is the minimum bandwidth of all possible bisection bandwidths (worst case bisectional configuration within one phase).

Source: `src/low_level/mpi`

Command Line:
```
$BENCH/bin/run -N $NODES -n $NPROCS -t 1 \
  PMB-EXT bidir_put bidir_get -multi 0 |
      tee mpi.onesided.out
```

Procedure: Divide all nodes of one installation phase into two equally sized sets, called “left” and “right” in the following.

Use as many processors as you need to reach the maximum aggregate bandwidth of the internal network, but at least one processor of each node must take part in the communication.

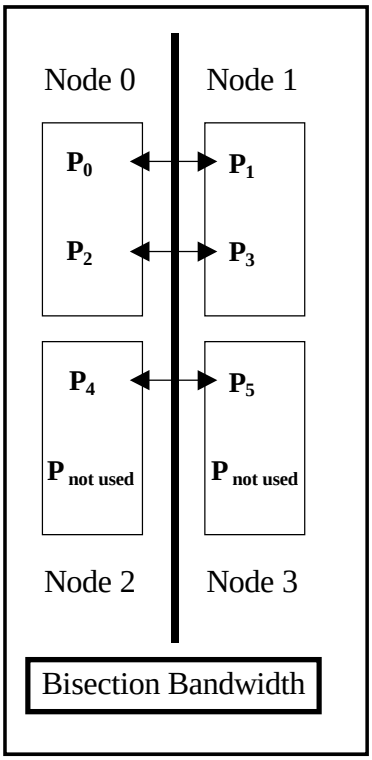
Run the benchmark with pairs of processors, where the two processors of a pair are on different sides of the configuration. All even numbered MPI processes must run in the "left" set, all odd numbered MPI processes must run in the "right" set of the system. This can easily be accomplished by using a hostfile or by using the subroutine readconfig.f and/or the configuration file `assign_nodes`.

The number of processes taking part in this benchmark run is denoted NPROCS.

The reported bandwidth must be scaled with the number of processor pairs which is half the number of MPI processes taking part in the benchmark.

aggregate bisection-bandwidth of a Phase =
(reported bandwidth) times (NPROCS/2)

Example: 4 nodes with 6 MPI processes take part in this example. The reported bandwidth of the benchmark has to be scaled by a factor of 3.



Results: Deliver mpi.onesided.out.

Take the bandwidth value for the case where NPROCS/2 groups are formed. The message size of 4 MByte (=4194304 Bytes) is used for the test cases. The measurements are done in aggregate and non-aggregate mode (as defined by the benchmark source code).

The highest bandwidth of the four cases specified above must meet the commitment by the vendor for the bandwidth and is multiplied with the number of processes taking part in the communication divided by 2.

Commitments and requirements:

Commitment:

Bandwidth for bidir_put (aggr. mode) * (NPROCS/2) = _____ GByte/s
Bandwidth for bidir_put (non aggr. mode) * (NPROCS/2) = _____ GByte/s
Bandwidth for bidir_get (aggr. mode) * (NPROCS/2) = _____ GByte/s
Bandwidth for bidir_get (non aggr. mode) * (NPROCS/2) = _____ GByte/s

(these four values are only for reference and qualitative evaluation)

Aggregate bisectional one-sided communication bandwidth _____ **GByte/s**

(Max. of above)

4.2 Storage subsystem benchmarks

4.2.1 IOBench 1: multi-stream read/write for SAN/DAS storage subsystem

Purpose: This benchmark tests the conventional I/O performance for the globally accessible parallel file systems (SAN/DAS) allowing concurrent I/O access. For this benchmark, each MPI process has its own local file pointer.

Source: `src/io_bench/iobench`

Procedure: The test is performed for Fortran and C by writing to and reading from a globally accessible file system in parallel by multiple processes from different nodes. Files are written and read in chunks of 1 MByte by default. Vendors may increase the chunksize up to 32MByte by changing the parameter CHUNKSIZE in iobench.F if this helps to increase the performance.

The size of each file must be greater than **three times the size of the memory of a node, divided by the number of MPI processes on that node.**

As many streams as needed to reach the required I/O bandwidth may be used.

One single file system should be used for the benchmark within Phase 1 or within Phase 2, respectively (20 GByte/s case).

Two different file systems may be used for benchmarking the combined Phase1 and Phase2 (40 GByte/s case), but the preferred way would be to write to and read from a single file system.

The program contains calls to close the files. **It must be ensured by the vendor that the data are really flushed to disk after closing the files. We have inserted a calls to FLUSH, but even this may not been sufficient.** The method used to achieve this must be disclosed to LRZ. It is not sufficient for this benchmark that the files or part of them are kept in memory only after closing the files. However, intermediate caching of data in memory in order to write/read larger chunks to/from disk is appreciated as it is an appropriate method for enhancing performance.

Modifications: `iobench.dat:` The file size (at least three times the memory on each node, divided by the number of processes on that node) and the paths and names of the files which are written to or read from must be specified in `iobench.dat`.

Calls for setting IO-buffers (e.g., “setvbuf”) or for flushing the data are permitted, as well as the use of run time settings. The insertion of compiler directives is permitted and appreciated. Other modifications of the remaining routines are not permitted.

Command lines: Run the benchmark with an increasing number of nodes and processors to show the scaling:

```
cat >iobench.dat <<EOD
100000   #Size of each file in MBytes (memory*3/procs per node)
5        #Number of iterations Do not change!
{globalfilename0}
{globalfilename1}
... etc

EOD
for NODES in 1 4 5 8 10 12 14 ...until saturation
do
  NPROCS=`expr $NODES \* proc_per_node`
  $BENCH/bin/run –N $NODES –n $NPROCS –t 1 \
      iobench <iobench.dat \
```

```
        >iobench.global.$NPROCS.out \
        2>iobench.global.$NPROCS.err
    mv iobench.res iobench.global.$NPROCS.res
done
```

Results: Deliver the output files `iobench.global.*.res`.

A detailed description of the I/O configuration must be given; particularly, all deviations from standard OS parameters must be fully disclosed.

The results are marked by the following lines:

```
Overall Results (aggreg. over all procs and all runs)
```

individually for Fortran and C, and for reading and writing.

The maximum I/O bandwidth for the offered system has to be reported together with the number of nodes and processors used.

Reference output: see subdirectory REFERENCE-OUTPUT.

Commitments and requirements:

Commitment:

I/O Bandwidth Reading (Fortran) = _____ GByte/s
I/O Bandwidth Reading (C) = _____ GByte/s
(these two values are only for reference and qualitative evaluation)

Number of Nodes: _____
Number of Processors: _____

I/O bandwidth for reading _____ **GByte/s**
(Max. of above)

I/O Bandwidth Writing (Fortran) = _____ GByte/s
I/O Bandwidth Writing (C)= _____ GByte/s
(these two values are only for reference and qualitative evaluation)

Number of Nodes: _____
Number of Processors: _____

I/O bandwidth for writing _____ **GByte/s**
(Max. of above)

I 63	The I/O bandwidth for reading as well as writing **to a single** file system on the SAN/DAS storage should be larger than **20 GByte/s**. for Phase 1 as well as for Phase 2. Check here if the requirement can be fulfilled: **[ ]**

I 64	For the combined Phases1+2, the I/O bandwidth for reading as well as that for writing using a single instance of the IObench program and employing at most **two file systems** for I/O should be larger than **40 GByte/s.** Check here if the requirement can be fulfilled: **[ ]**

I 65:	It should be possible to access all relevant parallel file systems on the SAN/DAS storage with approximately equal efficiency from both Phases.

This was already set forth as a requirement in section 2.2.3:

4.2.2 IOBench 2: multi-stream read/write for NAS storage subsystem

Purpose: This benchmark tests the conventional I/O performance for **a single** shared file system (NAS part of the disks). For this benchmark, each MPI process has its own local file pointer.

Source: `src/io_bench/iobench`

Procedure: as described in section 4.2.1

Modifications: as described in section 4.2.1

command lines: as described in section 4.2.1

Results: as described in section 4.2.1

Commitments and requirements:

Commitment:

I/O Bandwidth Reading (Fortran) = _____ GByte/s
I/O Bandwidth Reading (C) = _____ GByte/s
(these two values are only for reference and qualitative evaluation)
Number of Nodes: _____
Number of Processors: _____

I/O bandwidth for reading _____ **GByte/s**
(Max. of above)

I/O Bandwidth Writing (Fortran) = _____ GByte/s
I/O Bandwidth Writing (C)= _____ GByte/s
(these two values are only for reference and qualitative evaluation)
Number of Nodes: _____
Number of Processors: _____

I/O bandwidth for writing _____ **GByte/s**
(Max. of above)

M 80: The I/O bandwidth of Phase 1 for reading as well as that for writing must exceed 600 MByte/s.

Check here if the requirement can be fulfilled: **[ ]**

M 81: The I/O bandwidth of the combined Phases 1+2 for reading as well as that for writing must exceed 800 MByte/s.

Check here if the requirement can be fulfilled: **[ ]**.

Reminder: It must be possible to access all relevant user file systems on the NAS storage with equal efficiency from both phases.

This was already set forth as a requirement in section 2.2.3.

4.2.3 MPI-IO Benchmark

Purpose: This benchmark tests the MPI-IO performance. The test is performed for C by writing to and reading from a single file in parallel with multiple processes from different nodes. The file is written and read in chunks of 32 MBytes.

Source: `src/io_bench/mpi_IO`

Procedure: The amount of data written by each node shall be at least **three times the memory of each node** of the offered configuration. This is controlled by the number of blocks given in the input file.

The programs contain a call to `MPI_File_close`. **It must be ensured by the vendor that the data are really flushed to disk** after the call to `MPI_File_close`. The method used to achieve this must be disclosed to LRZ. It is not sufficient for this benchmark that the file or part of it are kept in memory only after the `MPI_File_close`. However, intermediate caching of data in memory in order to write/read larger chunks to/from disk is appreciated as it is an appropriate method for enhancing performance.

Run the benchmark with an increasing number of nodes to show the scaling.

```
for NODES in 1 4 5 8 10 12 16 … until the required
                                  bandwidth is reached
do
  NPROCS=`expr $NODES \* $proc_per_node`
  $BENCH/bin/run –N $NODES –n $NPROCS –t 1 ./mpi_IO \
            >mpi_IO.out
  cp mpi_IO.out mpi_IO.$NPROCS.out
done
```

Modifications: `mpi_IO.dat:`

```
Directory /xxxx/yyyy/        #Directory where files
                             are written and read
blocksize 32                 #chunksize 32MB.
                             Do not change!
nblocks 32                   #Number of blocks
                             written and read by
                             each process.
                             Must be changed.
nretries 5                   #The measurement is
                             repeated five times.
                             Do not change!
```

The insertion of compiler directives is permitted and appreciated.

Calls for setting keys (`MPI_Info_set`) and flushing the data are permitted. Other modifications of the remaining routines are not permitted.

Results: Deliver the output files `mpi_IO.*.out`.

A detailed description of the I/O configuration must be given; particularly, all deviations from standard OS parameters must be fully disclosed.

Reference output: see subdirectory REFERENCE-OUTPUT.

Commitments and requirements:

Commitment:

Number of Nodes: _____
Number of Processors: _____

Commitment for the MPI-IO bandwidth for reading _____ GByte/s

Number of Nodes: _____
Number of Processors: _____

Commitment for the MPI-IO bandwidth for writing _____ GByte/s

I 66: The MPI-IO bandwidth for reading as well as for writing should be larger than 5 GByte/s for each Phase.

Check here if the requirement can be fulfilled: **[ ]**

I 67: MPI-IO bandwidth for reading as well as for writing using **at most two instances** of a program should be larger than 10 GByte/s for the combined Phases 1+2.

Check here if the requirement can be fulfilled: **[ ]**

4.2.4 Metadata Benchmark

Purpose: This benchmark tests the transaction rates of the SAN/DAS and NAS file systems. Therefore it creates, reads, writes and deletes a configurable number of files of a configurable range of sizes.

The incidence of each transaction type and its affected files are chosen randomly to minimize the influence of file system caching, file read ahead, disk level caching, and track buffering.

The benchmark is a parallel version of the Postmark benchmark.

Source: `src/io_bench/metabench`

Procedure: Run the benchmark with a total number of **32 MPI processes**.

The placement for processes on the nodes should be round-robin, i.e. MPI process 0 runs on Node 0, process 1 runs on Node 1, …, Process 4 runs on Node 0, etc.

Run the benchmark in both the SAN/DAS and the NAS file systems.

Command Line:

```
cd {SAN/DAS filesystem}
$BENCH/bin/run –N $NODES –n 32 –t 1 metabench
              >metabench.SAN.out

cd {SAN/DAS filesystem}
$BENCH/bin/run –N $NODES –n 32 –t 1   metabench
              >metabench.NAS.out
```

Modifications: No modifications, except for the placement of processes, are allowed.

Results: `metabench.SAN.out` and `metabench.NAS.out` must be delivered.

A detailed description of the I/O configuration must be given; in particular, all deviations from standard OS parameters must be fully disclosed.

Reference output: see subdirectory REFERENCE-OUTPUT.

Commitments for Phase 1: (these values are only for reference and qualitative evaluation)		
Number of MPI Processes	Number of Transactions/second in SAN/DAS Filesystem	Number of Transactions/second in NAS Filesystem
1		
2		
4		
8		
16		
32		

M 82: On the SAN/DAS storage subsystem for each Phase,

at least 200 transactions/sec must be achieved using one MPI process,

at least 800 transactions/sec must be achieved with 32 MPI processes running simultaneously.

Check here if the requirement can be fulfilled: **[ ]**

M 83: On the NAS storage subsystem,

at least 400 transactions/sec must be achieved using one MPI process,

at least 5000 transactions/sec must be achieved with 32 MPI processes running simultaneously.

Check here if the requirement can be fulfilled: **[ ]**

4.3 Kernel Benchmarks

4.3.1 RINF1

This benchmark tests the floating point and integer performance of various one-dimensional loop kernels; depending on the access pattern various aspects of the processor architecture and the memory hierarchy are tested. This benchmark is derived from the RINF1 Genesis benchmark originally developed in the HPC Department at the University of Southampton, England.

Case 2 of this benchmark is very important, as it is used for the scaling of the interconnect and memory bandwidth. LRZ considers the result of this benchmark as a first indication of the sustainable performance of the system.

Purpose: Of the many loop types implemented, two (double precision and long integer triads) contribute to the **quantitative** evaluation of the bandwidths, and one (random read) contributes to the quantitative evaluation of the latencies within the memory hierarchy. All other loop types are only relevant for **qualitative** assessment of the node architecture. The cases relevant for the quantitative evaluation are

Case 2: Double Precision Vector Triad

```
A(I) = B(I) * C(I) + D(I)
```

evaluated as 3 loads, 1 store and 2 floating-point operations.

Case 13: Long Integer Vector Triad

```
IA(I) = IB(I) * IC(I) + ID(I)
```

evaluated as 3 loads, 1 store and 2 integer operations.

Case 38: Latency of memory system using random access on integer array

```
K = IA(I); I = K
```

evaluated formally as one integer operation (although only 1 integer load is actually involved). For this case **only the single-threaded results** are of interest

Source:. `src/low_level/rinf`

Procedure: The program needs to run **on all CPUs of each installation phase**. Multiple copies of the program are started via MPI, and each MPI process spawns OpenMP threads as determined by the OMP_NUM_THREADS variable. Performance values are determined by running with up to 32 threads, or the maximum number of CPUs available in the shared memory node, whichever is smaller. For the **bandwidth** measurements, a logarithmically integrated average is formed for vector lengths starting with 100000 * OMP_NUM_THREADS, and ranging up to 16 Million. Hence, cache-based systems gain a **thread-cumulative** advantage if the cache size is larger than ~3.2 MBytes, while vector systems should be able to perform well throughout all relevant vector lengths. For the **latency** measurements, only a serial run is required.

Please consult the **README** file in the source directory for instructions on how to modify the sources and build the executables for the various runs to be performed.

Command Lines:

```
NCPU={number of CPUs of a node)
TOTALNODES={number of nodes in a Phase}
for OMP_NUM_THREADS  in 1 2 4 8 16 32
do
  # number of processes of one Node
  NPROCS1=`expr $NCPUS / $OMP_NUM_THREADS`
  # total number of processes
  NPROCS=`expr $NPROCS1 \* $TOTALNODES`
  $BENCH/bin/run –N $TOTALNODES –n $NPROCS \
     –t OMP_NUM_THREADS rinf1.exe
done
```

Modifications: Allowed modifications include:

- Choice of optimal compiler flags
- Optimization directives in source code
- Memory padding to be configured during setup
- Memory Layout by suitable arrangement of arrays in COMMON block
- Choice of Fortran77, Fortran90 or C version of code for optimal performance

Results: For each of the three cases (2, 13, 38) above, take the highest aggregate performance achieved by using 8, 16 or 32 threads per MPI process with Fortran77, Fortran90 or C to calculate the per-CPU performance (you need to divide the printed out number - which refers to possibly multi-threaded MPI processes - by the number of threads used by each MPI

process). If the number of CPUs of a node of your system is smaller than 32 leave the corresponding values below blank.

Submission of **4 thread results** is only allowed if the case 2 per-CPU performance while running 4 threads per MPI process **is above 1.5 GFlop/s.**

Reference output: see subdirectory REFERENCE-OUTPUT.

Commitments and Requirements:

Note: Divide results from program run by the number of threads to obtain per-CPU performance numbers!

Case 2: Double Precision Triad, 4 Threads (see above)	_____	GFlop/s
Case 2: Double Precision Triad, 8 Threads	_____	GFlop/s
Case 2: Double Precision Triad, 16 Threads	_____	GFlop/s
Case 2: Double Precision Triad, 32 Threads	_____	GFlop/s
Maximum **per-CPU** performance T1 (This value - which must exceed 200 MFlops - is important for further evaluation)	_____	GFlop/s
Case 13: Long Integer Triad, 4 Threads (see above)	_____	GIop/s
Case 13: Long Integer Triad, 8 Threads	_____	GIop/s
Case 13: Long Integer Triad, 16 Threads	_____	GIop/s
Case 13: Long Integer Triad, 32 Threads	_____	GIop/s
Maximum **per-CPU** performance T2 (these values are only for reference and qualitative evaluation)	_____	GIop/s
Case 38: Latency Random Read, serial run	_____	GIop/s
Maximum **per-CPU** performance T3 (these values are only for reference and qualitative evaluation)	_____	GIop/s
Weighted per-CPU performance (0.7*T1+0.2*T2+0.1*T3)	_____	**GFlop/s**

Commitment:

Aggregate compute performance (as described in 3.2) = (Weighted per-CPU performance) times (Number of CPUs of that installation phase) =	_____	GFlop/s

Commitment:

Case 2 performance (Double Precision Triad) **per Node**
with the provision that the per-CPU performance
exceeds 200 MFlop/s. _____ GFlop/s

(this value is used for the scaling of the memory bandwidth and of the interconnect, see 2.2.5 and 2.3.2)

4.3.2 FFT

Purpose: Fast Fourier Transform is required by many scientific applications; hence an efficient implementation of one- and higher dimensional FFT routines is mandatory on any modern HPC system. The implementation, beyond being thread-safe, should also support multi-threading with reasonably good scalability. From the hardware point of view, many scatter-gather operations are performed which – depending on the length of the transform – exert considerable stress on the memory subsystem. The following cases are run:

Case 1: One-dimensional complex-to-complex transform using single precision complex data. This is provided as reference source code.

Case 2: One-dimensional complex-to-complex transform using double precision complex data. This is provided as reference source code.

Case 3: One-dimensional complex-to-complex transform using double precision complex data based on the Temperton library, which is also provided as source code.

Case 4: One-dimensional complex-to-complex transform using single precision complex data and **a vendor-provided** library routine.

Case 5: One-dimensional complex-to-complex transform using double precision complex data and **a vendor-provided** library routine.

Case 6: Three-dimensional complex-to-complex transform using double precision complex data and the reference library routine provided as source.

Case 7: Three-dimensional complex-to-complex transform using double precision complex data and **a vendor-provided** library routine.

Only the cases **4, 5, and 7** are relevant for obtaining performance numbers.

Source: `src/low_level/fft`

Procedure: The program needs to run on all CPUs of each installation phase. Multiple copies of the program are started via MPI, and each MPI process spawns OpenMP threads as determined by the OMP_NUM_THREADS variable. Performance values are determined by running with up to 32 threads, or the maximum number of CPUs available in the shared memory node, whichever is smaller. Performance averages are obtained by running transform lengths from 512 up to 2097152 in powers of two. For three-dimensional transforms the lengths are factored in various ways starting off with 8 x 8 x 8.

Please consult the **README** file in the source directory for instructions on how to modify the sources and build the executables for the various runs to be performed.

Command Lines: For obtaining the Case 4 and 5 performance numbers please run

```
NCPU={number of CPUs of a node)
TOTALNODES={number of nodes in a Phase}
for OMP_NUM_THREADS  in 1 2 4 8 16 32
do
```

```
  # number of processes of one Node
  NPROCS1=`expr $NCPUS / $OMP_NUM_THREADS`
  # total number of processes
  NPROCS=`expr $NPROCS1 \* $TOTALNODES
  $BENCH/bin/run –N $TOTALNODES –n $NPROCS` \
     –t OMP_NUM_THREADS rfft.exe  (or rfft3d.exe)
done
```

For the three-dimensional transform (Case 7), please replace **rfft.exe** by **rfft3d.exe** in the above script.

Modifications: Allowed modifications include:

- Choice of optimal compiler flags
- Memory padding to be configured during setup
- Choice of optimal library routine, capable of doing efficiently multi-threaded complex and double complex FFT in one and three dimensions, respectively

Results: Please deliver all result files rfft*.res.

For the Cases 4, 5, and 7, take the highest aggregate performance achieved by using 8, 16 or 32 threads per MPI process. If the number of CPUs of a node of your system is smaller than 32 leave the corresponding values below blank. For filling in the performance numbers in the tables below please calculate the **complete node performance** by scaling, e.g., for the 8 thread case on a 32 way node you need to run four MPI processes and multiply the printed out performance results by four.

Submission of **4 thread results** is only allowed if one MPI process with 4 threads yields an average performance of **above 5 GFlop/s for case 5.**

Reference output: is available in the subdirectory `results`.

Commitments and Requirements:

Commitments:

For filling in the performance numbers, calculate the complete node performance!

Case 4: Single Precision 1D FFT, 4 Threads (see above)	_____	GFlop/s
Case 4: Single Precision 1D FFT, 8 Threads	_____	GFlop/s
Case 4: Single Precision 1D FFT, 16 Threads	_____	GFlop/s
Case 4: Single Precision 1D FFT, 32 Threads	_____	GFlop/s
Maximum **Node** Performance T1	_____	GFlop/s
Case 5: Double Precision 1D FFT, 4 Threads (see above)	_____	GFlop/s
Case 5: Double Precision 1D FFT, 8 Threads	_____	GFlop/s
Case 5: Double Precision 1D FFT, 16 Threads	_____	GFlop/s
Case 5: Double Precision 1D FFT, 32 Threads	_____	GFlop/s
Maximum **Node** Performance T2	_____	GFlop/s
Case 7: Double Precision 3D FFT, 4 Threads (see above)	_____	GFlop/s
Case 7: Double Precision 3D FFT, 8 Threads	_____	GFlop/s
Case 7: Double Precision 3D FFT, 16 Threads	_____	GFlop/s
Case 7: Double Precision 3D FFT, 32 Threads	_____	GFlop/s
Maximum **Node** Performance T3	_____	GFlop/s
Average per-Node Performance (T1+T2+T3)/3 = (must be larger than 4.0 GFlop/s)	_____	GFlop/s
Aggregate compute performance (as described in 3.2), with arbitrary number of threads, **with the provision that the average performance per node is larger than 4.0 GFlop/s =** (Average per node) times (number of nodes) =	_____	**GFlop/s**

4.3.3 SipSolver

Procedure: A strongly-implicit procedure (SIP) solver is used. This method is suitable for solving systems of linear equations resulting from a discretization of partial differential equations. It is widely used in fluid mechanics and therefore of great importance. Regarding the implementation, different approaches are possible. For details see:

http://www.lrz.de/services/compute/hlrb/optvecpar/optimization_for_CFD.pdf.[11]

The current benchmark is intended to test the saturated single node performance and the scaling behavior.

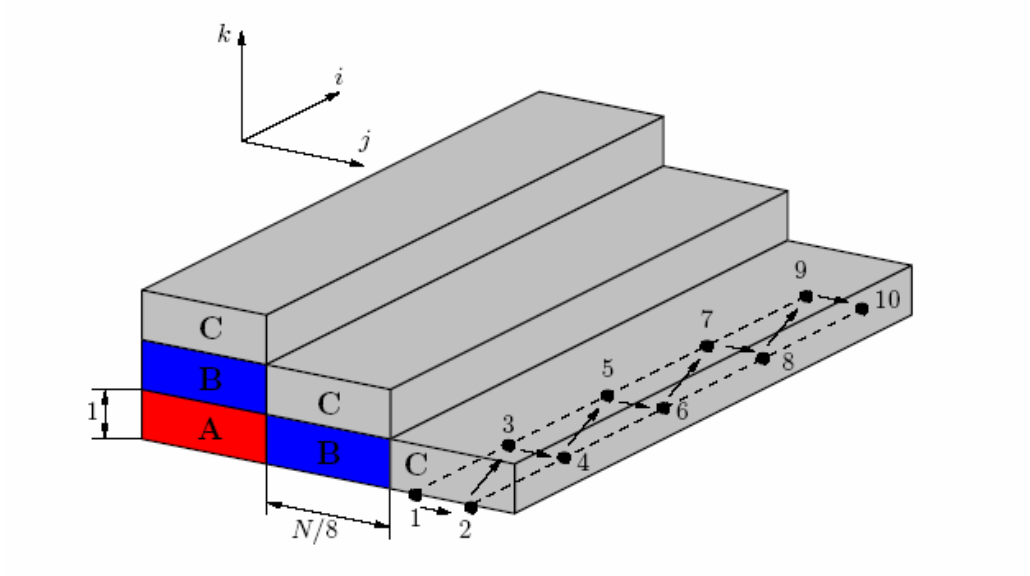
Currently we have implemented the following six versions:

Case 101 (SipThreeDSolver): a straightforward way is to iterate over all nodes (i,j,k) in 3 do-loops.

[11] Paper by the HPC group of Regionales Rechenzentrum Erlangen

Case 102 (SipThreeDSolver_regopt): same as case 101, but one loop is split in multiple parts to improve register usage.

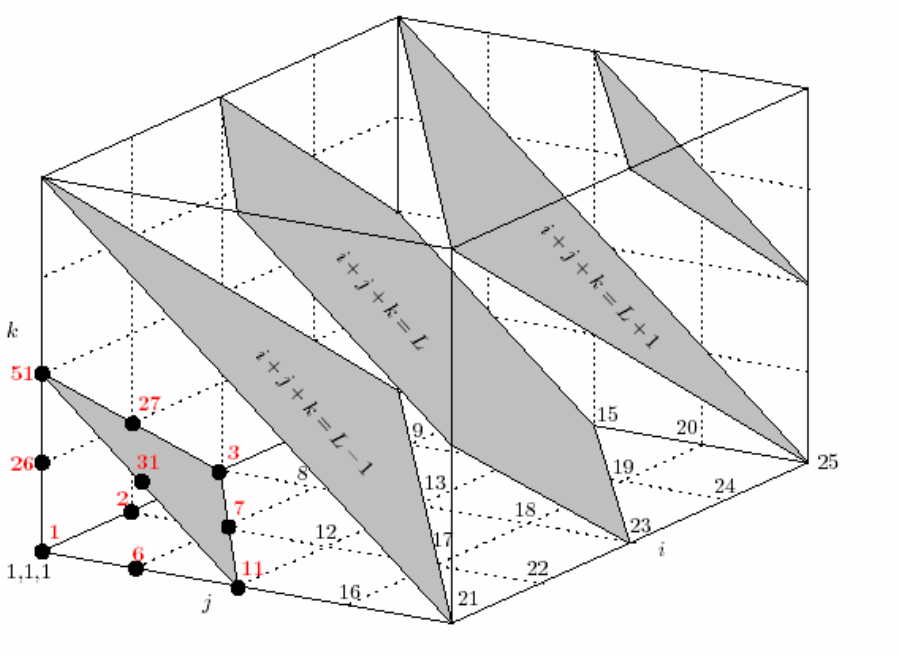
Case 103 (SipThreeDSolver_ppp, pipeline parallel processing): Data dependencies may prohibit automatic parallelization of the 3D-Version of the SIP-Solver. However, the Fortran90 compiler on the Hitachi SR8000 is able to resolve the dependencies with its specific pipeline parallel processing technique. This technique is re-programmed here with explicit OpenMP statements. The system is divided up into chunks of a certain size (see the following figure). Parallelization is applied to the loop along the *j*-direction (the middle loop). Calculation of any chunk is delayed by a barrier until the chunk left of it has been processed. Consequently, blocks with equal colour in the figure are calculated concurrently. This leads to load imbalance in the “wind-up” and ”wind-down” phases of this pipeline since some CPUs have to wait for the first few chunks to be calculated; this effect is negligible for a sufficiently large lattice.



Case 104 (SipHyperplaneSolver): A hyperplane is defined as

$$L = i + j + k = \text{const.}$$

Considering the rules for the LU-decomposition, for a given index triple (i, j, k) in hyperplane L only data from hyperplane L-1 are required because there is only one coordinate changing at a time (access to (i-1, j, k), (i, j-1, k) and (i, j, k-1)). Therefore, calculations on data within a plane are independent of each other and can be parallelized. However, as a result of the dependency of L on L-1, hyperplanes have to be calculated one after another. The following figure shows how the hyperplanes are arranged within the cube.

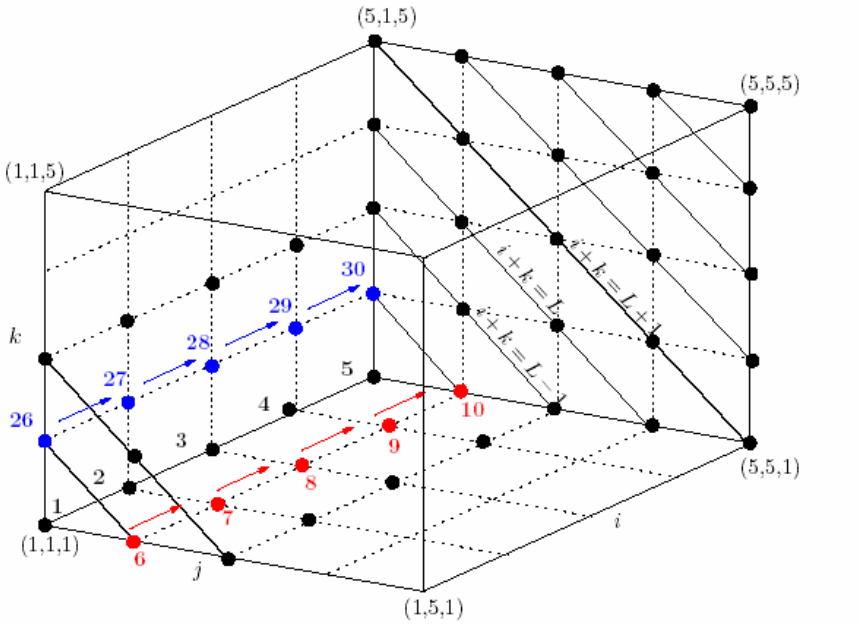


Case 105 (SipHyperplaneSolver_sr8k): pipeline-parallel variant of case 104 optimized for Hitachi SR8000.

Case 106 (SipHyperLineSolver): Similar to hyperplanes one can define hyperlines for which

$$L = j + k = \text{const.}$$

There is no need for a data point on one diagonal to access another one on the same diagonal. Therefore, within a hyperline the lines in i-direction can be processed in parallel.



Hints: For getting good performance, it may be necessary to ensure that the data layout in the initialization phase is the same as during the computa-

tion phase. For SMP systems the allocation policy (e.g., First Touch) determines the appropriate initialization strategy.

Source: `src/low_level/rinf2`

Procedure: Always run the benchmark **on as many as possible CPUs** of an installation phase according to the methods described in 3.2. For this purpose multiple copies of the executable are started. Ideally, the number of threads times the number of MPI processes is equal to the number of CPUs of a particular installation phase.

Run the benchmark with an increasing number of threads until the number of CPUs of a node or 32 threads are reached.

Command Lines:

```
NCPU={number of CPUs of a node}
TOTALNODES={number of nodes in a Phase}
for OMP_NUM_THREADS  in 1 2 4 8 16 32
do
  # number of processes of one Node
  NPROCS1=`expr $NCPUS / $OMP_NUM_THREADS`
  # total number of processes
  NPROCS=`expr $NPROCS1 \* $TOTALNODES`
  $BENCH/bin/run –N $TOTALNODES –n $NPROCS \
     –t OMP_NUM_THREADS rinf2 <Sipbench.301.dat
done
```

Modification: To avoid implications from memory layout on the performance, the problem size may be chosen by the vendor in the range between 295 and 305. The default problem size in the input file was set to 301.

Reference outputs: see directory REFERENCE-OUTPUT.

These reference outputs refer to the internal control output which is contained for the actual run in `rinf2.testoutput`.

Results: Take the highest performance of the six code versions.

In the box below, please enter the performance numbers for single MPI processes (running on a **filled node**, e. g. 4 MPI processes with 4 threads each on a 16-way node). Use at most 32 threads per MPI process.

The maximum value of the various methods is taken as the performance commitment for each multithreading configuration. From these values, compute the aggregate performance according to the methods described in section 3.2, while ensuring the required minimum performance.

If the number of CPUs of a node of your system is smaller than 32 leave the corresponding values below blank.

Commitments and requirements:

Commitments for Phase 1:

Case	Threads:	1	2	4	8	16	32	
101: Sip3DSolver		_____	_____	_____	_____	_____	_____	GFlop/s
102: Sip3DSolver_regopt		_____	_____	_____	_____	_____	_____	GFlop/s
103: Sip3DSolver_ppp		_____	_____	_____	_____	_____	_____	GFlop/s
104: SipHyperPlSolver		_____	_____	_____	_____	_____	_____	GFlop/s
105: SipHyperPlSolver_sr8k		_____	_____	_____	_____	_____	_____	GFlop/s
106: SipHyperLineSolver		_____	_____	_____	_____	_____	_____	GFlop/s

(these values are only for reference and qualitative evaluation)

Performance **per MPI process** = _____ _____ _____ _____ _____ _____ GFlop/s
(max of above, choose one with > 4 GFlop/s for aggregate compute performance)

Aggregate compute performance (as described in 3.2),
with the provision that the performance per MPI process exceeds 4 GFlop/s: _____**GFlop/s**

4.3.4 ZHEEVD

The LAPACK routine ZHEEVD computes all the eigenvalues, and optionally all the eigenvectors, of a complex hermitean matrix. It uses a divide and conquer algorithm to compute eigenvalues and – if requested – the eigenvectors.

Purpose: Evaluate the (saturated) single node performance and scaling behavior

Source: `src/low_level/rinf2`

Procedure: Always run the benchmark on all CPUs and nodes of an installation phase. For this purpose multiple copies of the executable are started. Ideally, the number of threads times the number of MPI processes is equal to the number of CPUs of a particular installation phase (see 3.2 for details).

Run the benchmark with an increasing number of threads until the number of CPUs of a node or 32 threads is reached.

Command Lines:

```
NCPU={number of CPUs of a node}
TOTALNODES={number of nodes in a Phase}
for OMP_NUM_THREADS  in 1 2 4 8 16 32
do
  # number of processes of one Node
  NPROCS1=`expr $NCPUS / $OMP_NUM_THREADS`
  # total number of processes
  NPROCS=`expr $NPROCS1 \* $TOTALNODES`
  $BENCH/bin/run –N $TOTALNODES –n $NPROCS \
     –t OMP_NUM_THREADS rinf2 <ZHEEVD.xxx.dat
done
```

Modifications: To avoid implications from memory layout on performance, the problem size may be chosen by the vendor in the range between 7995 and 8005. The default problem size in the input file was set to 8001. If the shared-memory scalability is bad, it is also allowed to internally change the algorithm to one with better scaling provided the numerical results can be reproduced with sufficient precision.

Reference outputs: see directory REFERENCE-OUTPUT.

These reference outputs refer to the internal control output which is contained for the actual run in `rinf2.testoutput`.

Results: In the box below, please enter the performance numbers for single MPI processes (running on a **filled node**, e. g. 4 MPI processes with 4 threads each on a 16-way node). Please use at most 32 threads per MPI process

Compute aggregate performance according to the methods described in 3.2. If the number of CPUs of a node of your system is smaller than 32 leave the corresponding values below blank.

Commitments and requirements:

Commitments for Phase 1:

Threads:	1	2	4	8	16	32	
Perf. **per MPI process**	_____	_____	_____	_____	_____	_____	GFlop/s

(choose one with > 8 GFlop/s for aggregate compute performance)

per-CPU performance =
(Performance per MPI process) / OMP_NUM_THREADS =

_____	_____	_____	_____	_____	_____	GFlop/s

Aggregate compute performance (as described in 3.2),
with the provision that the performance per MPI process is greater than 8 GFlop/s:
(per-CPU performance) * (number of CPUs offered) _____ **GFlop/s**

4.3.5 DMRG

The Density-Matrix Renormalization Group Algorithm (DMRG) is used for solving quantum problems in solid state physics and theoretical chemistry. It is an alternative to expensive (resource-consuming) exact diagonalization[12].

The core of DMRG is a sparse matrix-vector multiplication (performed within the Davidson diagonalization). The main matrices and vectors are decomposed into dense matrices of smaller size such that the dominant operation at the lowest level is a dense matrix-matrix multiply. In this way optimized Level 3 BLAS routines can be used.

The parallelization approach implies the use of SMP-parallel BLAS (no code changes) and OpenMP parallelization for sparse matrix-vector multiplication.

Purpose: Evaluate the performance of a C++ code on a single node using OpenMP.

Case dmrg_bench1: input file for cache-bound computation (short runtime ~ 1 min)

Case dmrg_bench2: input file for cache-bound computation (long runtime ~1 hour)

Case ddmrg_bench : input file for memory-bound computation (runtime ~ 10-30 min)

Source: `src/apps/DMRG`

[12] Parallelization strategies for density matrix renormalization group algorithms on shared-memory systems. G. Hager, E. Jeckelmann, H. Fehske, G. Wellein, Journ. Comp. Phys. 2003.

Procedure: The benchmark program is to be run on a single node only.

Run the benchmark with an increasing number of threads until the number of CPUs of the node or 8 threads are reached. For more than 8 threads, DMRG is not expected to scale sufficiently well. Note that multiple copies of the program must be started as long as a single instance of the program does not fill the node.

Command Lines:

```
cd run
NCPU={number of CPUs of a node}
TOTALNODES={number of nodes in a Phase}
for OMP_NUM_THREADS in 1 2 4 8
do
  # number of processes of one Node
  NPROCS1=`expr $NCPUS / $OMP_NUM_THREADS`
  # total number of processes
  NPROCS=`expr $NPROCS1 * $TOTALNODES`
  for INPUT in dmrg_bench1 dmrg_bench2 ddmrg_bench
  do
    $BENCH/run –N $TOTALNODES –n $NPROCS \
       –t OMP_NUM_THREADS dmrg <${INPUT}.in
  done
done
```

Modifications: Instead of matrix/fastblas1.h, linkage with a proprietary BLAS library is permitted (and encouraged) if that library is compatible with OpenMP in the outer loops of the sparse matrix multiplications. I.e., the proprietary library should perform serial computations inside OpenMP-parallel regions, and switch to parallel computation within the otherwise serial parts of the program. Especially, the DGEMM routine used on long, narrow matrices has considerable impact on the DMRG performance.

Reference outputs: see directory: run/REFERENCE_OUTPUT

Results: Take the average of the performance values of the three program runs. The program reports these values to stdout in the section "CPU time profiles", line “PERFORMANCE”. The values are the number of necessary floating point operations (given in the first line of each input file) divided by the wallclock runtime. The number of floating point operations determined on an IBM p690 are as follows:

Case dmrg_bench1:	12908247.266 Mflop
Case dmrg_bench2:	14362.188 Mflop
Case ddmrg_bench:	613853.141 Mflop

Compute aggregate performance according to the methods described in section 3.2.

Commitments and Requirements:

Commitments for Phase 1:

Case	Threads:	1	2	4	8	
dmrg_bench1		_____	_____	_____	_____	GFlop/s
dmrg_bench2		_____	_____	_____	_____	GFlop/s
ddmrg_bench		_____	_____	_____	_____	GFlop/s

(these values are only for reference and qualitative evaluation)

Average per MPI process =
(dmrg_bench1+ dmrg_bench2+ ddmrg_bench)/3=
_____ _____ _____ _____ GFlop/s
(choose one with > 0.5 GFlop/s for aggregate compute performance)

Aggregate compute performance (as described in 3.2), =
(performance of one CPU) * (Number of CPUs of that installation phase),
with the provision that the performance
per MPI process is greater than 0.5 GFlop/s: _____ **GFlop/s**

4.3.6 Laser

The laser simulation code is supposed to test whether the C++ compiler is capable of translating complicated template constructions resulting from the concept of *expression templates* [1,2].

The simulation of lasers leads to a non-linear Helmholtz eigenvalue problem. In laser simulation, this equation is approximately solved by a Gauss-mode analysis or beam propagation method. Those approximations are not very accurate in some applications. Therefore, this simulation code tries to approximate the equation directly. To this end, very fine grids are necessary for resolving the small wavelength of the solution of this equation.

The code is based on the C++ library EXTEMP for solving linear and non-linear partial differential equations on general 3D domains. This library normally runs very efficiently even on vector-like machines - which is not self-evident for C++ code - thanks to the extensive use of expression templates.

The expression template technique is described in

[1] T. Veldhuizen: "Expression Templates". In: C++ Report, Vol. 7 No. 5 June 1995, pp. 26-31. Online version see reference [3].

[2] T. Veldhuizen: Techniques for Scientific C++. Indiana University Computer Science Technical Report #542, August 2000. Online version see reference [3].

[3] http://osl.iu.edu/~tveldhui/papers

[4] http://www.oonumerics.org

Purpose: This benchmark consists of two parts:

laser/Expr: The short program `extemp` is used for determining the **single-CPU performance** of code generated by the C++ compiler from template expressions. `extemp` adds and multiplies vector components of arrays of lengths 1,000, 10,000, 100,000, and 1,000,000. The compiler must be able to evaluate expression templates up to a depth of 13.

laser/Laser: The complete simulation program **`laser`** will demonstrate whether the compiler is able to compile more complicated expression template constructions. **No performance measurements** are required, but the runtime of the resulting code must not exceed 15 minutes.

Source: `src/apps/laser/Expr`

`src/apps/laser/Laser`

Procedure: Run the benchmark program `extemp` on a single processor or alternatively, if automatic parallelization is available and if an improved aggregate compute performance (as described in 3.2) can be achieved, on more processors.

Compile and run the program `laser` without and including compiler optimization flags.

Command Lines:

```
cd Expr
NCPU={number of CPUs of a node}
TOTALNODES={number of nodes in a Phase}
OMP_NUM_THREADS={optimal number of threads, e.g. 1}

$BENCH/bin/run –N $TOTALNODES –n $NPROCS \
       –t OMP_NUM_THREADS extemp
cd -
cd Laser
time gmake all
time gmake run
```

Modifications: Allowed modifications are described in section 3.1.2.

Especially, the source code lines marked with "benchmark core" in `Expr/extemp.cc` must not be modified manually in order to exploit the Horner scheme. However, we would like the compiler to recognize this Horner scheme. Then the resulting code only needs to perform 12 instead of 27 floating point operations per vector component and iteration. The scheme itself is rated with 12 floating point operations.

Reference output: see directory REFERENCE_OUTPUT

Results: Report the performance values that the program `extemp` prints to stdout.

Report whether your C++ compiler is able to compile laser/Laser, and if yes, please give the duration of the entire compilation.

Commitments and Requirements:

Commitments for Phase 1:

Program extemp (from laser/Expr):
Performance of one process reported by `extemp` _____ GFlop/s

Aggregate compute performance, (as described in 3.2),
with the provision that the performance per CPU is greater than 0.5 GFlop/s: _____ GFlop/s

Commitments for Phase 1:

Program laser (from laser/Laser)
Compile time of program `laser` **without** optimization: _____ seconds
Compile time of program `laser` **with** optimization: _____ seconds

I 68: Execution time of program `laser` (from laser/Laser) with an arbitrary number of threads should complete in less than 900 seconds.

Check here if the requirement can be fulfilled: **[ ]**

4.3.7 LINPACK

The standard parallel LINPACK benchmark will be used to obtain an estimate of the peak performance of the system; this benchmark also gives an impression of the quality of the vendor's BLAS implementation. It is the only benchmark that will run on the combined Phase 1 and Phase 2 system. The following aspects of the system will be tested:

Case 1: Peak performance of Phase 1 and Phase 2 separately

Case 2: Peak performance of Phase 1 and Phase 2 coupled

For a given problem size N, αN^2 bytes of memory storage will be required; ideally, α should not be much larger than 8 if eight-byte floating-point words are used. If this storage is distributed across P (shared-memory) nodes, the amount of memory per node required will be

$$M = \frac{\alpha N^2}{P}.$$

For execution of Case 1 above, the largest reasonable problem size should be used[13]. For execution of Case 2 above, the problem size needs to be chosen so as to achieve the requirements given below. For all measurements, the performance L(P, N) is determined by

$$L(P,N) = \frac{\frac{2}{3}N^3 + 2N^2}{T_{measured}},$$

where $T_{measured}$ is the execution time for problem size N and node count P.

Source: No source is provided for this benchmark. The vendor can either use his own implementation or make use of the HPL sources publicly available from Netlib at http://www.netlib.org/benchmark/hpl. The method, source code and configuration used should be disclosed, especially any differences to the publicly available methods.

Procedure: **Case 1:** Choose the problem size N for optimal total performance and measure the execution time.

Case 2: Choose the problem size $N_{coupled}$ so as to fulfill the requirements for the coupled Phase 1 and 2 system and measure the execution time.

Configuration: This benchmark can be optimized by the vendor as he sees fit. The following considerations apply, however:

1. Optimization of the communication pattern. If MPI is not used for communication, please describe your communication method and give the reason for not using MPI. Please also describe your process layout, block size and solution method.
2. Optimized BLAS. Please describe what variant of BLAS is used. Note that DGEMM must preserve the operation count. **Hence, use of e.g., the Winograd-Strassen algorithm is prohibited.**

[13] For a system with 20 TFlop/s peak performance and 10 TBytes of memory used at an efficiency of >80% the run time would be around 10 hours. So it may be reasonable to run at only half memory filling to obtain run times on the order of 1-2 hours.

3. MPP vs. Hybrid. You can run this benchmark either in MPP or in hybrid mode, whatever yields the best results. Please provide details on your parallelization method, especially whether multi-threaded MPI calls are used.

Results: For quantitative evaluation, only the Case 1 results for the maximum performance (per Phase) are relevant. All other cases only require commitments; additional minimum requirements are listed below.

Commitments and Requirements:

Commitment for Phase 1:	
Size of LINPACK =	_____
Time =	_____ seconds
LINPACK Performance Phase1	_____ **GFlop/s**

Commitment for combined Phases 1+2:	
Size of LINPACK =	_____
Time =	_____ seconds
LINPACK Performance of combined Phases 1+2	_____ **GFlop/s**

M 84 The LINPACK Performance of the combined Phases 1+2 must exceed the LINPACK Performance of Phase 1 by a factor of at least 1.5.

4.4 Application Benchmarks

4.4.1 BEST

BEST (TRATS) is a CFD code for solving 2D/3D flows utilizing a Lattice Boltzmann BGK model. The code has been developed at the LSTM Institute of the University Erlangen.

Source: `src/apps/best`

Executable: trats_<name_of_architecture>

Modifications: Porting to and optimization for your hardware may be required.

Internally, the code uses MPI_DIMS_CREATE and MPI_CART_CREATE to create the layout of the processors. It is recommended to specify "nproc=0,0,0" in the config file for automatic segmentation. With some MPI implementations, however, better performance might be obtained by manually providing the segmentation. If the latter is the case, these performance results may be used.

Building: Since this code has been ported to most common HPC architectures, it deploys its own system of makefiles, which are tailored for the specific architecture.

The makefile system allows to set specific options; these are set for the benchmark and must not be changed; they are defined by –D defines and are passed to the build process; the makefile will then source a system-specific makefile, in which compilers, loaders, flags and options are set. Use these system specific make.includes as a guideline for your architecture.

Multithreading: Multi-Threading is permitted and may be useful. Please report the number of nodes, processes and threads which are used.

Procedure: Compile the code after modifying the appropriate make.include. The code has calls to a graphics library which is available for several architectures. For this benchmark, it can be exchanged by a dummy library, which has dummy entries for the required calls. Source for such a dummy library can be found in `benchmark/lib/tecplot (dummy.f)`.

There are several subdirectories N<num> in the directory `runs`, <num> denoting the number of MPI processes. Within these directories run the code with the respective number of MPI-processes. The input data files are called `trats.cfg` (the code expects the input data in this file and therefore does not need an extra argument.). Data for 1, 2, 4, 8, 16, 24, and 32 processor runs are supplied. The cases for more processes can easily be constructed by inspecting the file `trats.cfg`. Lines 2 and 3 (the values `,glob,nx,ny,nz` and `,nproc,PE_X,PE_y,PE_z`) need to be adjusted. Make sure that `nx` is a multiple (factor is `PE_X`) of the corresponding value for the single-process case (`nx=256`).

Run at least the 1, 16, 32, and 64 process setups. If 128 CPUs are available, please also provide data for this problem size.

Top Performance: In order to obtain the target performance your proposed system can deliver, you need to construct your own adapted test case. Take the input data for a single process as a template, exchanging the number for processes in x-direction with your chosen number and adjusting the problem size by multiplying the number of grid points in x-direction with the chosen number of processes (see also the README file in the `best` base directory).

Results: The performance of the code is obtained by inspecting the standard output and denoting the achieved “MLUPS” rate (5 MLUPS being roughly equivalent to 1 GFlops/s). For each test case run, supply the output of the program and fill in the chart with the respective numbers.

Commitments and requirements:

Commitments for Phase 1: (these values are only for reference and qualitative evaluation)				
Run #	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]
1		1		
2		16		
3		32		
4		64		
5		128		
$T_{>1000\ GFlop/s}$				

Commitment for Phase 1:

Aggregate compute performance (as described in 3.2),
with the provision that the performance of a single instance of BEST is greater than 1000 GFlop/s: _____ GFlop/s

4.4.2 BQCD

BQCD (Berlin QCD Program) is a hybrid Monte-Carlo program that simulates Quantum Chromodynamics with dynamical standard Wilson fermions.

The computations take place on a four-dimensional regular grid with periodic boundary conditions. The updates are local (i.e., only nearest neighbors are needed).

The kernel of the program is a standard conjugate gradient solver with even/odd pre-conditioning. As a consequence all arrays are stored in an even/odd ordered fashion and the four indices are collapsed into a single one. The access to neighbors is handled by lists.

The parallelization is done by a regular grid decomposition in the highest 3 dimensions. The values from the boundaries of the neighboring processors are stored in the same array as the local values. The local values have indices 1, ..., volume/2. The boundary values have indices > volume/2.

The memory for the arrays is dynamically allocated during initialization.

Apart from rounding errors the program gives identical results for any grid decomposition.

The total domain size in the example input files is: 64 x 64 x 64 x 64.

You may increase the domain size to improve scalability.

Source: `src/apps/BQCD/Apps/BQCD`

Executable: `bqcd`

Modifications: Porting to and optimization for your hardware may be required.

In addition to the usual standard optimization techniques mentioned in 3.1.2 you are allowed to do the following modifications.

Routines d() and d_dag():

The following modifications to these routines are allowed:

- the sequence of floating point operations may be changed according to the rules of algebra
- the sequence of communication operations may be changed as long as remote data is available at the right point in time
- the data layout may be changed
- the subroutine structure within d() and d_dag() may be modified
- Fortran, C, C++ may be used as programming languages
- any communication library may be used

If you change the data layout you have to make sure that outside cg() the original data structures can be used, i.e., at the beginning of cg(), data would have to be copied from the original data structures to the new ones and at the end of cg() data would have to be copied back correspondingly. At a few locations outside of cg() the subroutines d() and d_dag() are called. Here necessary copy operations are also permitted.

Multithreading: Multithreading is permitted and may be useful. Please report the number of nodes, processes and threads which are used.

Procedure: The input for the benchmark run is stored in the file `input.BENCH`. In `input.BENCH` the domain size is defined in the line:

```
64 64 64 64 LX LY LZ LT
```

The definitive number of MPI processes per y-, z- and t-direction has to be defined in the line:

```
1 yy zz tt NPEX NPEY NPEZ NPET
```

The total number of MPI processes is `yy * zz * tt`.

yy, zz, tt must be dividers of LY, LZ, LT respectively.

The domain size may be increased to improve performance and scalability.

It is not mandatory that LX, LY, LZ and LT be powers of two. They may be changed to fit to the number of processors available for benchmarking, e.g., 64 64 48 80.

Run the program with an increasing number of processes and/or threads (starting from the lowest possible number) up to a sufficiently large number to show the scaling behavior, subject to the following conditions:

At least five runs with varying numbers of processes and/or threads must be performed to demonstrate the scaling behavior. The domain size (`LY * LZ * LT`) has to be increased in proportion to the number of processors (`yy * zz * tt`).

A performance *Perf(CG)* value of at least 800 GFlop/s must be reached in at least one of these runs. Since we evaluate the aggregate performance of the total system, it is probably best to run the program with the lowest possible number of processors necessary to achieve the required performance.

Command Line: Prepare at least five input data sets with the desired settings for the layout and number of the MPI-Processes and threads

```
$BENCH/bin/run –N … –n … –t …  \
     bqcd < input.BENCH.1 >out.BENCH.1
```

```
$BENCH/bin/run –N … –n … –t …  \
     bqcd < input.BENCH.2 >out.BENCH.2
...
```

Testing**:** Smaller test cases may be generated by modifying `LX`, `LY`, `LZ`, `LT` in `input.BENCH`.

Examples: Reference input and output can be found in the directory `Reference`.

Results: The total performance of the cg-solver can be obtained by running the script `summary` on the output produced by `bqcd`.

Please deliver all result files `out.BENCH.*`

Commitments and requirements:

Commitments for Phase 1: (these values are only for reference and qualitative evaluation)				
Run #	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]
1				
2				
3				
4				
5				
$T_{>800\ GFlop/s}$				

Commitment for Phase 1:

Aggregate compute performance (as described in 3.2), **with the provision that the performance of a single instance of bqcd is greater than 800 GFlop/s** **_____ GFlop/s**

4.4.3 Cactus

Cactus is an open source problem solving environment designed for scientists and engineers. Its modular structure easily enables parallel computation across different architectures and collaborative code development between different groups.

Cactus originated in the academic research community, where it has been developed and used over many years by a large international collaboration of physicists and computational scientists.

The computations of the subset of cactus chosen for this benchmark take place on a 3-dimensional grid.

Source: `src/apps/cactus`

Executable: `cactus_bench-`*`machine`*

Modifications: Porting to and optimization for your hardware may be required. Please refer to the Cactus Users Guide (which can be generated by doing a `make UsersGuide` in the cactus source directory) for options to include

specific libraries (like LAPACK or MPI). We have included a PostScript version of the Users Guide and other guides which might be useful in the `doc` directory.

Multithreading: Multithreading is permitted and may be useful. Please report the number of nodes, processes and threads used for running the program.

Procedure: The input for the final benchmark is stored in the file `LRZBench.par` in

`Cactus/arrangements/CactusBench/BenchAdm/bench`.

The problem size is determined by the parameter `global_nsize` (respectively `local_nsize`).

By defining `local_nsize=XXX` the problem is kept constant in size (and work) per process independent of the number of processes used.

This option is used for weak scaling (iso-granular scaling).

The chosen size of `local_nsize=172` will require about 3 GByte of local memory per process.

By defining `global_nsize=XXX` the problem is kept constant in size (and work) per run. This option is used for speed-up tests.

Please run the following tests for the benchmark evaluation:

a) **speedup tests**: with parameter `global_nsize = 172`, run with 4, 8, 16 and 32 processes.

b) **iso-granular tests**: with parameter `local_nsize = 172`, run with 8, 16, 32, 64 and (if possible) 128 processes.

c) **target performance**: a performance value of at least **500 GFlop/s** must be reached in at least one run of the iso-granular tests. You might need more than 128 processes to achieve the target performance; indicate this in the table below.

Command Line:
```
mpirun –np $NPROCS \
      cactus_bench-machine LRZBench.par \
            >BENCH.$NPROCS.out
```

Testing: Smaller test cases may be generated by modifying `local_nsize` or `global_nsize` in `LRZBench.par`. A script `gflops1` is provided in the subdirectory `utils` which extracts the performance information from the output file (given as argument).

Examples: Reference input and output can be found in the directory `Cactus/arrangements/CactusBench/BenchADM/LRZ.test`. Here you find the test case `Test_reference.par`, the output file generated with a single processor run and the directory `Test_Reference_Dir` with several output files. Do a `diff` on the file `ADM_gxx_3D_diagonal.xg` to determine whether your reference run generated the same results (this case was run on IA64 and SR8000 and produced identical results).

Results: Please deliver all result files `BENCH.*`.

Commitments and requirements:

Commitments for Phase 1: (these values are only for reference and qualitative evaluation)				
Run #	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]
a 1		4		
a 2		8		
a 3		16		
a 4		32		
b 1		8		
b 2		16		
b 3		32		
b 4		64		
b 5		128		
c >500 GFlop/s				

Commitment for Phase 1:
Aggregate compute performance (as described in 3.2), **with the provision that the performance of a single instance of cactus is greater than 500 GFlop/s** **_____ GFlop/s**

4.4.4 HEPFP

This benchmark derives from the ab-initio treatment of the excitation and ionization of highly excited Rydberg atoms subject to intense electromagnetic fields. Rydberg atoms under external driving in the microwave regime represent one of the very few realistic and experimentally accessible realizations of time dependent, classically chaotic systems, and therefore allow for the direct search for the "fingerprints of chaos" in quantum transport phenomena.

The solution of the exact quantum mechanical eigenvalue problem (including the field induced coupling to the atomic continuum without approximation) requires the diagonalization of banded, complex symmetric matrices with large dimensions und storage requirements.

Source: `src/apps/hepfp`

Executable: `hepfp`

Multithreading: Multithreading is permitted and may be useful. Please report the number of nodes, processes and threads which are used.

Procedure: Run the program with an increasing number of processes and/or threads, starting from the lowest possible number up to a sufficiently large number to show the scaling behavior, subject to the following conditions:

- At least five runs with different numbers of processes and/or threads must be performed to demonstrate the scaling behavior

- A performance value of at least **120 GFlop/s** must be reached in at least one of these runs. Since we evaluate the aggregate performance of the total system, it is probably best to run the program with the lowest possible number of processes necessary to achieve the required performance.

```
$BENCH/bin/run –N … -n … -t … \
        hepfp <hepfp.dat >hepfp.out.1

$BENCH/bin/run –N … -n … -t … \
        hepfp <hepfp.dat >hepfp.out.2
```

etc. for five different sets of processes/nodes and or threads.

Testing**:** Smaller test cases are contained in input1.dat, input2.dat, …, input6.dat.

Results: Please deliver the result files `hepfp.out.*`.

The requested performance values are contained in the last lines of the result files and are marked by the tag `GFLOP/s`. Note that for an eigenvalue problem of the size solved here the precision of the eigenvalues will be limited to 7-8 significant figures; any deviation within this limit will be acceptable.

Commitments and requirements

Commitments for Phase 1: (the values are only for reference and qualitative evaluation)				
Run #	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]
1				
2				
3				
4				
5				
$T_{>120\ GFlop/s}$				

Commitment for Phase 1:

Aggregate compute performance (as described in 3.2),
with **the provision that the performance of**
a single instance of hepfp is greater than 120 GFlop/s **_____ GFlop/s**

4.4.5 MGLET

MGLET is a code for direct numerical simulation of the Navier-Stokes equations for incompressible flows. It uses an explicit time step (leapfrog) for the propagation of the momentum equation and a SIP (ILU) solver for the solution of the Poisson equation for the pressure.

Source: `src/apps/mglet`

Executable: `mglet.exe`

Modifications: Porting to and optimization for your hardware may be required.

Building: **All REAL statements are to be interpreted as REAL*8 for this particular benchmark!**

Make sure that you compile for 64-bit REALs (e.g., by using suitable compiler switches like –r8 supported by the Intel or IBM Compiler).

Since this code has been ported to most common HPC architectures, it deploys its own system of makefiles, which are tailored to the specific architectures.

Use these system-specific makefiles as a guideline for building the code.

The code needs to be rebuilt for each different input deck, because it defines array sizes statically. In practice this means that only the main routine `mlet.src` must be recompiled since it INCLUDEs the problem-dependent dimensions file.

To simplify the process, input data and parameter files for the required runs are supplied.

We supply a tool `setparams` in the subdirectory `utils` that can be used to generate new input data sets as well as the necessary include files necessary for the right dimensioning of the code for the respective input data.

Procedure: For testing and verification purposes we supply the datasets

`test_02.np2.reference.dat,`

`test_02.np4.reference.dat` and

`test_02.np6.nx_2.ny_3.reference.dat`

and the corresponding result files.

Please run the following tests for the benchmark evaluation:

a) Speedup tests (strong scaling)

Construct a test case with IMX, JMX, KMX of 1284, 244, 404 and run it with 8, 16 and 32 processes.

The required memory for the run is roughly 28 GBytes (as measured on IA-64). If you cannot meet the memory/process requirement, you may double the number of processes, use processors on other nodes leaving some processors empty, or use the hybrid programming model with more than one thread per MPI process.

Test cases and the corresponding include files are supplied.

Report the synthetic GFlop/s number which is printed at the end of the output.

b) Iso-granular tests (weak scaling)

Construct a test case with IMX, JMX, KMX of 260, 484, 404 and run it with 4 processes.

Set the first dimension IMX to (256*2)+4=516 for 8 processes. In this fashion construct test cases for 16, 32, 64 and 128 processes (for the case with 128 processes IMX is then 8196 etc.) The required memory is roughly 2.9 GBytes per process (as measured on IA-64) for all cases.

If you cannot meet the memory/process requirement, you may use processors on other nodes leaving some processors empty, or use the hybrid programming model with more than one thread per MPI process.

Test cases and the corresponding include file for 4 processes are supplied.

c) Weak scaling, Iso-granular test with arbitrary geometry

A performance of more than 500 GFlop/s has to be reached with an arbitrary number of processes and threads.
In order to demonstrate this performance, you may need to construct your own adapted test case.
Using the tool `setparams` this task should not be too difficult. It may be advisable to use the largest case of the iso-granular test cases as a starting point.

Performance: Performance of the code is printed at the end of the run in the output. Report this synthetic GFlops number.

Multithreading: Multithreading is permitted and may be useful, particularly to keep the problem size per MPI process small. Please report the number of nodes, processes and threads which are used.

Commitments and requirements:

Commitments for Phase 1:
(these values are only for reference and qualitative evaluation)

Run #	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]
a 1		8		
a 2		16		
a 3		32		
b 1		4		
b 2		16		
b 3		64		
b 4		128		
c >500 GFlop/s				

Commitment for Phase 1:

Aggregate compute performance (as described in 3.2)**,**
for an arbitrary test case (weak scaling),
with the provision that the performance of a
single instance of mglet is greater than 500 GFlop/s: _____ GFlop/s

4.4.6 NWChem

NWChem is a computational chemistry program developed for use on parallel high performance computers. NWChem is provided by the Pacific Northwest National Laboratory. While a large spectrum of quantum chemical and molecular dynamics methods is available in NWChem, only a single method has been selected for this benchmark.

NWChem uses Global Arrays for communication. Detailed descriptions of NWChem and Global Arrays are available on the PNNL web site under the URL http://www.emsl.pnl.gov/docs/nwchem/nwchem.html .

Source: The source code is available at the above mentioned web site. The source code is free of charge, but you have to register to use the software, and you will have to sign a license agreement that will allow you free use of the program for your purposes but prevent you from distributing copies of it. Please contact PNNL for the source code of NWChem since LRZ does not have the right to provide you with the source code.
Only NWChem version 4.5 may be used for this benchmark. You are free to use whatever version of Global Arrays that works properly on your system. For internal testing at LRZ, Global Arrays version 3.2 was used.

Executable: You need to build the executable for your hardware yourself.

Modifications: Porting to and optimization for your hardware may be required.

In addition to the usual standard optimization techniques mentioned in 3.1.2 you are allowed to do the following modifications:

You are free to modify the way the SCF computation is run by adding a line like
`semidirect filesize 0 memsize 75000000`
to the input file with values for `filesize` and `memsize` optimally suited for your system.
Furthermore, you are free to modify the memory usage, i.e., to modify the line
`memory stack 100 mb heap 100 mb global 1500 mb`
of the input file.
You may **not** change the geometry, the basis set or the method itself (e.g., to CCSD instead of CCSD(T)).
All modifications have to be disclosed.

Procedure: The input for the final benchmark is stored in the file `nwchem-benchmark.nw`.
Make sure that no data (except the input file) from a previous run are present in the working directory.

```
rm benchmark.*
```

(If the results of a previous, successful run are present in the files `benchmark.*`, the CCSD calculation will stop after 2 iterations, while it will need 13 iterations when started without restart information.)

Run the benchmark using

```
$BENCH/bin/run –N … -n … -t …nwchem \
   nwchem-benchmark.nw > nwchem-benchmark.nw.out
```

Performance: Divide the predetermined number of floating point operations ($9.24 * 10^{14}$ FP Operations = 924 Tera Floating Point Operations) by the wall time as delivered by the above command, i. e.,

Performance in GFlop/s= 924000 / (time in seconds)

A minimum performance value of at least 100 GFlop/s must be achieved in this benchmark.

Examples: An example output can be found in the file `nwchem-benchmark.nw.out`; a smaller test example input file (`nwchem-example.nw`) as well as reference output is available as well.

The energy of the CCSD(T) computation has to agree up to the eighth digit behind the decimal point. Thus, executing the command

```
grep "Total CCSD(T) energy:" \
     nwchem-benchmark.nw.out
```

should deliver an output like

```
Total CCSD(T) energy:    -911.905574626632642
```

Anything like

```
Total CCSD(T) energy:    -911.90557462....
```

is acceptable. With the example input, anything like

```
Total CCSD(T) energy:    -900.43697011....
```

is acceptable.

Commitments and requirements:

Commitments for Phase 1: (these values are only for reference and qualitative evaluation)				
Time [s]	Number of Nodes	Number of MPI Processes	Number of Threads	Performance [GFlop/s]

Commitment for Phase 1:
Aggregate compute performance (as described in 3.2), **with the provision that the performance of a single instance of NWChem is greater than 100 GFlop/s _____ GFlop/s**

5 Benchmark Evaluation

For determination of the quantitatively best offer the considerations of section 5.3 are applied. The final ranking of the offers is obtained after taking into account the qualitative considerations described in section 5.4. The winning system must then achieve the weighted overall performance as described in sections 5.1 and 5.2 during the acceptance tests for Phase 1 and Phase 2, respectively.

5.1 Weighted overall performance of Phase 1

The weighted overall compute performance of Phase 1 is defined as the weighted mean of the following benchmarks:

Benchmark	Weight Factor	Aggregate Compute Performance [GFlop/s] as defined by 3.2	Aggregate Compute Performance times Weight Factor [GFlop/s]	See chapter
Kernels (together 40%)				
Rinf	0.09			4.3.1
FFT	0.06			4.3.2
SipSolver	0.06			4.3.3
ZHEEVD	0.05			4.3.4
DMRG	0.05			4.3.5
Laser	0.04			4.3.6
LINPACK	0.05			4.3.7
Applications (together 60%)				
BEST	0.13			4.4.1
BQCD	0.11			4.4.2
Cactus	0.09			4.4.3
HEPFP	0.09			4.4.4
MGLET	0.11			4.4.5
NWChem	0.07			4.4.6
Weighted Overall Performance (GFlop/s)	1.00	---	_____	---

Table 1: Weighted overall computing performance

Commitment for phase 1
Weighted overall performance of Phase 1 _____ GFlop/s

The overall computing performance of Phase 1 does not enter into the ranking scheme directly, since we compare each benchmark individually (see 5.3). However,

M 85: During the **acceptance test for Phase 1** the performance of the individual benchmarks and the overall performance have to be demonstrated. The performance of individual benchmarks may fail to reach the committed values by a margin of 5% but the commitments for the overall compute performance for phase 1 as defined here must be reached or exceeded.

5.2 Overall performance of Phase 2

It is the intention of LRZ that the vendor offers the same performance for the additional nodes of Phase 2 as for Phase 1, i.e., the performance of the combined Phases 1+2 doubles compared to Phase 1.

The overall compute performance of the additional nodes of Phase 2 is defined as the weighted mean of the benchmarks, run on the nodes of Phase 2. The vendor must give a commitment here for the overall performance of Phase 2. Typically, the vendor will give here the same commitment as for Phase 1.

The vendor may recompile and/or relink the benchmark programs or insert new directives, but no major changes of the program codes are allowed, compared to Phase 1.

I 69 The weighted overall performance **of Phase 2** should be nearly equal to the weighted overall performance of Phase 1, but should not be less than the weighted overall performance of Phase 1.

M 86: During the **acceptance test for Phase 2** the performance of the individual benchmarks and the overall performance have to be demonstrated. The performance of individual benchmarks may fail to reach the committed values by a margin of 10% but the commitments for the overall compute performance for Phase 2 as defined here must be reached or exceeded.

Commitment for Phase 2
Weighted overall performance of Phase 2: _____ GFlop/s

5.3 Procedure for ranking the performance of the offered systems

The procedure aims at ranking the offered systems according to their performance and evaluating their relative performance when compared with each other. This goal is ensured by the following procedure:

First, LRZ will check whether the **minimum requirements** and **exclusion criteria** are met.

Subsequently, the performance of the systems will be evaluated based on the committed performance of all compute nodes.

Only the committed benchmark results for Phase 1 and the LINPACK for the combined Phases 1+2 will be used in the ranking procedure, since we expect that the overall performance of the two phases is nearly equal.

To standardize the different characteristics of the individual benchmarks, the aggregate performance values for each single benchmark will be expressed separately as the ratio with respect to the best-performing system for this benchmark:

$$V = P / P_{best}$$

Thus, numerical values between 0 and 1 are obtained for each individual benchmark program. The ratios V that result from the individual benchmarks and the combined LINPACK will be multiplied by the weight factors g stipulated for the benchmarks in Table 2 and subsequently summed up.

$$R = \sum_i V_i * g_i$$

R is denoted as the ranking number. The numerical values of R also range between 0 and 1. R is considered to be a measure for the relative strength of an offered system with respect to all offered systems.

Benchmark	Short summary of requirements	Weight g (%)	Section
MPI (total weight 15 %)			
The interconnect balance for MPI_THREAD_SERIAL	----	1	4.1.1
Interconnect balance of a node	> 0.4 (Byte/s) / (Flop/s)	3	4.1.2
Aggregate bisectional bandwidth within a Phase	----	2	4.1.3
MPI Latency within a Phase	< 8 µs	3	4.1.3
MPI Latency between the two Phases	< 10 µs	2	4.1.4
Aggregate bisectional bandwidth between Phase 1 and Phase 2	> 2 * 40 GByte/s	3	4.1.4
Bisection Bandwidth for one-sided Communication	----	1	4.1.5
I/O (total weight 10 %)			
Aggregate I/O Performance for SAN/DAS	>20 GByte/s in Phase 1, >40 GByte/s for combined Phases 1 and 2	6	4.2.1
MPI-IO	>5 GByte/s in Phase 1, >10 GByte/s for combined Phases 1 and 2	2	4.2.3
Metabench for SAN/DAS	>200 for one MPI process	1	4.2.4
Metabench for SAN/DAS	>800 for 32 MPI processes	1	4.2.4
Kernels (total weight 30 %)			
Rinf	>200 MFlop/s per CPU	7	4.3.1
FFT	>4 GFlop/s per Node	4	4.3.2
SipBench	>4 GFlop/s per MPI Process	4	4.3.3
ZHEEVD	>8 GFlop/s per MPI Process	3	4.3.4
DMRG	>0.5 GFlop/s per MPI process	3	4.3.5
LASER	>0.5 GFlop/s per CPU	3	4.3.6
LINPACK Phase 1	----	3	4.3.7
LINPACK combined Phase 1 + Phase 2	> 1.5 times LINPACK Phase 1	3	4.3.7
Applications (total weight 45%)			
BEST	>1000 GFlop/s for the single application	10	4.4.1
BQCD	>800 GFlop/s for the single application	8	4.4.2
Cactus	>500 GFlop/s for the single application	7	4.4.3
HEPFP	>120 GFlop/s for the single application	7	4.4.4
MGLET	>500 GFlop/s for the single application	8	4.4.5
NWChem	>100 GFlop/s for the single application	5	4.4.6

Table 2: Weights and requirements for the performance evaluation scheme

5.4 Qualitative evaluation

To obtain a qualitative overall impression of the system all benchmarks as well as other aspects of the offer will be examined for conspicuous characteristics that may have an impact on the achievable computing performance or the usability and manageablity of the system.

For characteristics that are striking in a positive or negative way, a corresponding evaluation weight will be assigned. This will usually happen only if a system differs substantially from the other systems or a clear failure to fulfill the requirements stated in this document is observed. Qualitative corrections of this kind are done very carefully by LRZ and will be justified and explained.

The examined characteristics are, in particular: scalability, possible bottlenecks of central memory access, the automatically achievable computing performance (through auto-parallelization/auto-vectorization), the internal interconnect as well as I/O behavior.

Less significant differences in computing performance will not be evaluated in this step, as this information is already contained in the benchmark results.

Examples for problems that may result in a negative qualitative correction to the benchmark performance include but are not restricted to:

- far too weak interconnect
- total breakdown of computing performance when performing out-of-cache-computations
- too extensive programming effort for porting and/or optimization
- insufficient scalability
- unbalanced system characteristics

Examples for items that may result in a positive qualitative correction include:

- outstanding computing performance
- outstanding computing performance in a benchmark program designated for qualitative evaluation
- excellent I/O performance
- outstanding automatically achievable computing performance
- excellent vendor support (e.g., on- and off-site staff provided by the vendor for hardware and software support)

Other criteria for upward or downward evaluation result from the criteria described in Chapter 2, and are described there.

The performance number R determined according to the prescription given in section 5.3, will be multiplied with a factor η in order to obtain the final qualified ranking number Q.

$$Q = R * \eta$$

This qualitative correction by LRZ is restricted to a range of $0.80 \leq \eta \leq 1.20$.

6 Table of key system parameters

Item and Unit	Phase 1	Phase 2	Combined Phases 1+2
Environment			
Footprint of the system incl. maintenance areas: length x width (m x m)			
Total Weight (kg)			
Specific weight per square meter (kg/m^2)			
Point Load (N)			
Ambient temperature (range) (°C)			
Relative humidity (range) (%)			
Air purity of computer room (µg/m^3)			
Cooling type (water or air)			
Inlet temperature (range) (°C)			
Outlet temperature (range) (°C)			
Expected electrical power consumption (kVA)			
Peak electrical power consumption (kVA)			
Expected heat emission into air (kWh)			
Expected heat emission into water (kWh)			
Peak heat emission into air (kWh)			
Peak heat emission into water (kWh)			
Voltage (V)			
Number of electrical phases			
Frequency of electrical current (Hz)			
Compute Nodes			
Number of nodes			
Number of processors per node			
Total number of processors			
Frequency of one processor (MHz)			
Peak number of instructions per second of one processor (Gip/s)			
Peak number of instructions per second of one node (Gip/s)			
Peak number of instructions per second of entire Phase 1 (Gip/s)			
Peak Floating point performance of one processor (only multiply/add, no div or sqrt) (GFlop/s)			
If applicable: Length of fixed point execution pipeline of processor for mutliply or add operations			
If applicable: Length of floating point execution pipeline of processor for multiply and/or add operations			
Peak Floating Point Performance of one Node (only multiply/add, no div or sqrt) (GFlop/s)			
Total Peak Floating Point Performance of entire Phase 1 (only multiply/add, no div or sqrt) (GFlop/s)			

If vector: Scalar peak floating point performance of one processor (GFlop/s)			
Size of memory of one node (GByte)			
Size of memory of entire Phase 1 (GByte)			
(Share of theoretical) Bandwidth to local memory of one Processor (GByte/s)			
Total Bandwidth to local memory of one node (GByte/s)			
Interleaving of memory access			
Total Bandwidth to local memory (GByte/s)			
If applicable: Size of L1 data cache (kByte)			
If applicable: Size of L1 instruction cache (kByte)			
If applicable: Size of L2 data cache (MByte)			
If applicable: Size of L3 data cache (MByte)			
If applicable: Bandwith of L1 data cache (GByte/s)			
If applicable: Bandwith of L2 data cache (GByte/s)			
If applicable: Bandwith of L3 data cache (GByte/s)			
If applicable: Cache line size of L1 data cache (Byte)			
If applicable: Cache line size of L2 data cache (Byte)			
If applicable: Cache line size of L3 data cache (Byte)			
If applicable: Size of data TLB (entries)			
If shared by more than one processor: Size of L2 data cache per processor (MByte)			
If shared by more than one processor: Size of L3 data cache per processor (MByte)			
Latency to local memory (Cycles)			
If applicable: Latency of L1 data cache (Cycles)			
If applicable: Latency of L2 data cache (Cycles)			
If applicable: Latency of L3 data cache (Cycles)			
If applicable: Latency of data TLB (Cycles)			
If applicable: Latency to CPU local memory (Cycles)			
If applicable: Latency to memory on remote node(s) (Cycles)			
If applicable: Associativity of L1 data cache			
If applicable: Associativity of L2 data cache			
If applicable: Associativity of L3 data cache			
If vector: Number of vector registers (per processor)			
If vector: Length of vector registers (per processor)			
If vector: Size of vector cache (MByte)			
Number of floating point registers (per processor)			
Size of floating point registers (Bit)			
Number of general purpose (integer) registers (per processor)			
Size of general purpose (integer) registers (Bit)			
If applicable: Number of predicate registers (per processor)			
If applicable: Size of predicate registers (Bit)			
If applicable: Number of branch registers (per processor)			

If applicable: Size of branch registers (Bit)			
If applicable: Number of control registers (per Processor)			
If applicable: Size of control registers (Bit)			
Maximum number of outstanding prefetches (per processor)			
Internal Interconnect			
Number of (bidirectional) links per node			
Theoretical bandwidth of one link (bidirectional) (GByte/s)			
Theoretical bandwidth of one node (bidirectional) (GByte/s)			
Theoretical bisection bandwith of the system (GByte/s)			
MPI latency (µs)			
Bidirectional bisection bandwidth between Phase 1 and Phase 2 (GByte/s)	---	---	
MPI latency between Phase 1 and Phase 2 (µs)	---	---	
External Interconnect			
Number of interfaces to the external network			
Aggregate bandwidth to the external network (GByte/s)			
Disk Storage			
Number of I/O nodes			
Size of SAN/DAS user storage (TByte)			
Size of NAS user storage (TByte)			
Aggregate theoretical bandwidth to/from SAN/DAS storage (GByte/s)			
Aggregate theoretical bandwidth to/from NAS storage (GByte/s)			
Capacity of disks containing the operating system, swap and/or paging space of the nodes per compute Node (GByte)			
Bandwidth to local disks (per node) (Mbyte/s)			