Solutions to Homework Problems in Chapter 2

Hwang, Fox and Dongarra: Distrubuted and Cloud Computing,

Morgan Kaufmann Publishers, copyright 2012

Note: The solutions of Chapter 2 problems were contributed by **Lizhong Chen, Siddharth Razdan.** and **Varun Palivela** of the University of Southern California and by **Xun Zhao** of Tsinghua University, under the supervision of **Kai Hwang** and **Yongwei Wu**, respectively.

Problem 2.1:

(a). Compact versus slack clusters:

In a compact cluster, the nodes are closely packaged in one or more racks sitting in a room, and the nodes are not attached to peripherals (monitors, keyboards, mice, etc). In a slack cluster, the nodes are attached to their usual peripherals (i.e, they are complete SMPs, workstations, and PCs), and they may be located in different rooms, different buildings, even geographically remote regions.

(b). Centralized versus decentralized clusters:

In a centralized cluster, all the nodes are owned, controlled, managed, and administered by a central operator. In a decentralized cluster, the nodes have individual owners.

(c). Homogeneous versus heterogeneous clusters:

A homogeneous cluster means that the nodes adopt the same platform. That is, they use the same processor architecture and the same operating system. Often, the nodes are from the same vendors. A heterogeneous cluster uses nodes of different platforms. Interoperability is an important issue in heterogeneous clusters.

(d). Enclosed versus exposed clusters

Intracluster communication can be either exposed or enclosed. In an exposed cluster, the communication paths among the nodes are exposed to the outside world. An outside machine can access the communication paths, and thus individual nodes, using standard protocols. In an enclosed cluster, intracluster communication is shielded from the outside world.

(e). Dedicated versus enterprise clusters:

A dedicated cluster has the following characteristics:

- It is typically installed in a desk side rack in a central computer room
- It is typically homogeneously configured with the same type of nodes
- It is managed by a single administrator group like a mainframe
- It is typically accessed via a front end system.

A dedicated cluster is installed, used and administered as a single machine.

- An enterprise cluster is mainly used to utilize idle resources in the nodes. It has the following characteristics:
- Each node is usually a full fledged SMP, workstation, or PC, with all necessary

peripherals attached

- The nodes are typically geographically distributed, not necessarily in the same room or even in the same building
- The nodes are individually owned by multiple owners. The cluster administrator has only limited control over nodes, as a node can be turned off at any time by its owner. The owner's "local" jobs have higher priority than enterprise jobs.
- The cluster is often configured with heterogeneous computer nodes. The nodes are often connected through a low-cost Ethernet

Problem 2.2:

(a) Availability without planned downtime.

Assume MTTF = One year = 365x24x60x60 = 31,536,000 seconds

Given MTTR = 10 = 30 = 40 seconds

Availability = MTTF/(MTTF + MTTR) = 31,536,000 / (31,536,000 + 40) = 99.99987%

(b). Availability with planned downtime for maintenance.

Now, MTTR = 40 + 1x60x60x52 = 187,200 seconds

Availability = MTTF/(MTTF + MTTR) = 31,536,000 / (31,536,000 + 187,200) = 99.89%

Problem 2.3:

a) Tianhe-1A

Architecture: The system consists of five major components. The compute subsystem houses all the CPUs and GPUs on 7,168 compute nodes. The service subsystem comprises eight operation nodes. The storage subsystem has a large number of shared disks. The monitoring and diagnosis subsystem is used for control and I/O operations. The communication subsystem is composed of switches for connecting to all functional subsystems.

Hardware components: This system is equipped with 14,336 six-core Xeon E5540/E5450 processors running 2.93 GHz with 7,168 NVIDIA Tesla M2050s. It has 7,168 compute nodes, each composed of two Intel Xeon X5670 (Westmere) processors at 2.93 GHz, six cores per socket, and one NVIDIA M2050 GPU connected via PCI-E. A blade has two nodes and is 2U in height (Figure 2.25). The complete system has 14,336 Intel sockets (Westmere) plus 7,168 NVIDIA Fermi boards plus 2,048 Galaxy sockets (the Galaxy processor-based nodes are used as frontend processing for the system).

A compute node has two Intel sockets plus a Fermi board plus 32 GB of memory. The operation nodes are composed of two 8-core Galaxy FT-1000 chips. These processors were designed by NUDT and run at 1 GHz. The theoretical peak for the eight-core chip is 8 Gflops/second. The complete system has 1,024 of these operational nodes with each having 32 GB of memory. It uses 7,168 compute nodes (with 448 CUDA cores/GPU/compute node) in parallel with 14,236 CPUs with six cores in four subsystems.

Operating system: It uses Kylin Linux, an operating system developed by NUDT and successfully approved by China's 863 Hi-tech Research and Development Program office in 2006. Kylin is based on Mach and FreeBSD, is compatible with other mainstream operating systems, and supports multiple microprocessors and computers of different structures. Kylin packages include standard open source and public packages, which have been brought onto

one system for easy installation.

Software support: The software stack on the Tianhe-1A is typical of any high-performance system. The NUDT builder developed a mathematics library, which is based on Intel's MKL 10.3.1.048 and BLAS for the GPU based on NVIDIA and optimized by NUDT. In addition, a High Productive Parallel Running Environment (HPPRE) was installed. This provides a parallel toolkit based on Eclipse, which is intended to integrate all the tools for editing, debugging, and performance analysis. In addition, the designers provide workflow support for Quality of Service (QoS) negotiations and resource reservations.

Parallelizing compilers: The system features FORTRAN, C, C++, and Java compilers from Intel (icc 11.1), CUDA, OpenMP, and MPI based on MPICH2 with custom GLEX (Galaxy Express) Channel support.

Packaging: The Tianhe-1A system is packaged in 112 compute cabinets, 12 storage cabinets, six communications cabinets, and eight I/O cabinets.

Cooling: The system has a footprint of 700 square meters and is cooled by a closed-coupled chilled water-cooling system with forced air. It takes about $20 million annually to run, maintain, and keep the system cool in normal operations.

New application: Most of them are specially tailored to satisfy China's national needs. (Parallel AMR method; Parallel eigenvalue problems; Parallel fast multipole methods; Parallel computing models; Gridmol computational chemistry; ScGrid middleware, grid portal; PSEPS parallel symmetric eigenvalue package solvers; FMM-radar fast multipole methods on radar cross sections; Transplant many open source software programs; Sandstorm prediction, climate modeling, EM scattering, or cosmology; CAD/CAE for automotive industry)

b) Strengths and Limitations :

The Tianhe-1A system demonstrated a sustained speed of 2.507 Pflops in Linpack Benchmark testing runs and thus became the No. 1 supercomputer in the 2010 Top 500 list. The software stack on the Tianhe-1A is typical of any high-performance system. It uses Kylin Linux, an operating system developed by NUDT and successfully approved by China's 863 Hi-tech Research and Development Program office in 2006.

The system has an efficiency of 54.58 percent, which is much lower than the 75 percent efficiency achieved by Jaguar and Roadrunner.

Problem 2.4:

1) Scalability definitions

a) Scalability over machine size

This indicates how well the performance of a scalable parallel computer will improve with additional processors. The resources increased are most frequently processors, but they could also be memory capacity and I/O capability. There also exist a maximum number of processors a system can accomodate and thus impose an upper bound of scalability over machine size.

b) Scalability over problem size

This indicates how well the syustem can handle larger problems with lager data size and workload. Apart from depending on machine size, it also depends on memory capacity, and communication capability of the machine.

c) Resource scalability

This refers to gaining higher performance or functionality by increasing the machine size (i.e.

the number of processors), investing in more storage (cache, main memory, disks), improving the software, etc. Within this dimension, three categories have to be considered. Machine size scalability indicates how well the performance will improve with additional processors. Scaling up in resource means gaining higher performance by investing more memory, bigger off-chip caches, bigger disks and so on. Finally, software scalability indicates how the performance of a system be improved by a newer version of the OS that has more functionalities, a better compiler with more efficient optimizations, more efficient mathematical and engineering libraries, more efficient and easy-to-use applications software and more user-friendly progrmming environment.

d) Generation scalability

This refers to the capability of a system to scal up the performance using the next generation components, such as a faster processor, a faster memory, a newer version of operating system, a more powerful compiler, etc, with the rest of the system be usable and modifiable as little as possible.

2) Three availability cluster configurations

Hot standby server clusters In a hot standby cluster, only the primary node is actively doing all the useful work normally. The standby node is powered on (hot) and running some monitoring programs to communicate heartbeat signals to check the status of the primary node, but is not actively running other useful workloads. The primary node must mirror any data to shared disk storage, which is accessible by the standby node using a second copy of data.

Active takeover clusters In this case, the architecture is symmetric among multiple server nodes. Both servers are primary, doing useful work normally. Both failover and failback are often supported on both server nodes. When a node fails, the user applications fail over to the available node in the cluster. Depending on the time required to implement the failover, users may experience some delays or may lose some data that was not saved in the last checkpoint.

Failover cluster This is probably the most important feature demanded in current clusters for commercial applications. When a component fails, this technique allows the remaining system to take over the services originally provided by the failed component. A failover mechanism must provide several functions, such as failure diagnosis, failure notification, and failure recovery. Failure diagnosis refers to the detection of a failure and the location of the failed component that caused the failure. A commonly used technique is heartbeat, whereby the cluster nodes send out a stream of heartbeat messages to one another. If the system does not receive the stream of heartbeat messages from a node, it can conclude that either the node or the network connection has failed.

Problem 2.5:

a) Various memory models.

UMA (Uniform Memory Access)

- All the processors in the UMA model share the physical memory uniformly.
- Access time to the memory location is independent of which processor makes the memory request or which memory chip contains the transferred data.
- Each processor may use a private cache.
- Suitable for general purpose and time-sharing applications by multiple users.
- It is less scalable than NUMA.

NUMA (Non Uniform Memory Access)

- Separate memory for each processor, to avoid the performance hit when several processors are trying to access the same part in memory.
- The memory access time depends on the memory chip position relative to the processor.
- A processor can access its own local memory faster than a non-local memory.
- Suitable for clusters.
- It is more scalable than UMA.
- Each address in the global space is typically assigned a fixed home node.

COMA (Cache only Memory Access)

- Uses local memory as cache in contrast to using it as the real memory as in the case of NUMA.
- There is no home node for a memory address. When a processor requests for memory the data may migrate.
- Compared to NUMA this reduces the number of redundant copies and hence utilizes the memory more efficiently.
- However it raises problems such as how to find a particular data and what to do if the local memory fills up.

DSM (Distributed Shared Memory)

- Here although the memories are physically separate they can be addressed as one (logically shared) address space.
- The address space is shared, two processors may reference the same location .
- Connection is relatively slow and loose (LAN, MAN)
- Can be page, variable or object based.

NORMA (No Remote Memory Access)

- Each processor has its own local memory that is not shared..
- They are the simplest to design and build and are currently used in supercomputers like Intel Paragon.
- A sequence of workstations on a local area network constitutes NORMA architecture.
- As opposed to NUMA architecture, NORMA has no hardware support for direct access to remote memory modules.
- They are more loosely coupled compared to NUMA machines.

b) Unique features of a cluster

- Clusters have a Single System Image (SSI). This means that by combining several systems in a cluster using SSI we get a single huge system..
- They have high availability with lots of redundancy in processors, memory, disks, I/O devices, networks and operating systems.
- They have high utilization as they use job management software to do load balancing, batching and parallel processing.
- They should be fault tolerant with checkpoint and rollback schemes.
- They should have a single control.
- They are symmetric i.e. a user can use a cluster from any node.
- They have a single file hierarchy.

c) Advantages of cluster over SMP server

- Can be built using commercially available off the shelf products.
- Can be easily scaled to meet the demand for more computing power.
- Higher price / performance ratio.

Problem 2.6:

This problem refers to Table 3.5, not Table 2.6. The COD and Violin are studied in Chapter 3, not in Chapter 2. Part (a) and (b) can be answered after studying Section 3.4 in p.155-168.

Problem 2.7:

We consider the top 10 systems in Top 500 list reported in Nov. 2011.

(a). The systems that have used x86 processers as follows : K computer, Cores: 705024; R_{max}: 10510Tflops; R_{peak}: 11280Tflops; Power: 12660KW. Cray XT5-HE Opteron: 224162; R_{max}: 1759Tflops; R_{peak}: 2331Tflops; Power: 6950KW. Cray XE6, Opteron 6136 8C, Cores: 142272; R_{max}: 1110Tflops; R_{peak}: 1366Tflops; Power: 3980KW. SGI Altix ICE 8200EX/8400EX, Cores: 111104; Rmax: 1088Tflops; R_{peak}: 1315Tflops; Power: 4102KW. Cray XE6, Opteron 6172 12C, Cores: 153408; R_{max}: 1054Tflops; R_{peak}: 1289Tflops; Power: 2910KW. Bull bullx super-node S6010/S6030, Cores: 138368; R_{max}: 1050Tflops; R_{peak}: 1254Tflops; Power: 4590KW. BladeCenter QS22/LS21 Cluster, Cores: 122400; R_{max}: 1042Tflops; R_{peak}: 1376Tflops; Power: 2345KW.

(b). The systems that have used GPUs to complement the x86 CPUs as follows: NUDT YH MPP, Dawning TC3600 Blade System, HP ProLiant SL390s G7.

Commodity GPUs are becoming high-performance accelerators for data-parallel computing. Modern GPU chips contain hundreds of processor cores per chip. Based on a 2010 report, each GPU chip is capable of achieving up to 1 Tflops for single-precision (SP) arithmetic, and more than 80 Gflops for double-precision (DP) calculations. Recent HPC-optimized GPUs contain up

to 4 GB of on-board memory, and are capable of sustaining memory bandwidths exceeding 100 GB/second.

The high performance of a GPU cluster is attributed mainly to its massively parallel multicore architecture, high throughput in multithreaded floating-point arithmetic, and significantly reduced time in massive data movement using large on-chip cache memory. In other words, GPU clusters already are more cost-effective than traditional CPU clusters. GPU clusters result in not only a quantum jump in speed performance, but also significantly reduced space, power, and cooling demands. A GPU cluster can operate with a reduced number of operating system images, compared with CPU-based clusters. These reductions in power, environment, and management complexity make GPU clusters very attractive for use in future HPC applications.

Problem 2.8:

(**a)** Wong and Franklin optimizes the checkpoint rate as $\alpha = \sqrt{}$ (o.5 $\phi\beta$), where ϕ is the mean rate of system failure and $1/\beta$ is the time needed to take a checkpoint. The time taken to execute the 1000 iterations without failure and checkpointing is 1000 x 10 x 3 = 30,000 minutes or 1.8×10^6 seconds. Since it is given that it fails once, ϕ is $1/1.8 \times 10^6$. During a checkpoint the values of arrays A, B and C must be saved along with all other parts of code. Hence the total memory required for a checkpoint is 120 x 3 + 16 = 376 MB

The disk read/write bandwidth is 1 Mb/second. Hence time required to save a checkpoint is 376 seconds. Also time required to restore from a checkpoint is 376 seconds. Therefore $1/\beta$ is 376 seconds. Replacing these values in the formula

$$\alpha = \sqrt{}\,[0.5 / (1.8 \times 10^6 \times 376)] = 2.7 \times 10^{-5}$$

Hence the optimum checkpoint interval is 36791.303 seconds or 613.188 minutes.

This means that since the program execution is 30,000 minutes checkpoint is saved 49 times which takes 376 seconds for each save. Also when the checkpoint is restored it takes another 376 seconds, total checkpoint saving and restoring time

= 376 x 49 + 376 = 18,800 seconds = 313.33 minutes

Also, after restoring it takes 0.5 x α time to recompute the values after failure which is 306.594 minutes. Therefore total delay is 619.924 minutes.

Hence worst-case execution time for completion of code is 30,619.924 minutes.

(b). In plain transparent checkpointing, when a checkpoint is saved the normal application operating is not stopped. Hence the only overhead is in restoring the checkpoint when a system failure occurs. Hence the total time required restoring the system after a failure is 376 seconds or 6.26667 minutes as shown earlier plus another 306.594 minutes to do the recomputation.

Therefore the worst-case execution time of the code is 30312.86 minutes.

(c). In forked checkpointing a child process is forked which creates a copy of the program's data space in memory while the parent process or the main program continues execution. However

this means that apart from the 376 MB required for the parent process we need another 376 MB for the child process. This means that a total of 752 MB needs to be saved in memory. However since we have only 512 MB it is a bad idea to use forked checkpointing.

(d). In user directed checkpointing we will save only the program data and not the other parts of the code. I.e. we will save only 360 MB and not the other 16 MB. Therefore it takes 360 seconds or 6 minutes. The part of the code where the user directives are added is after

C = hoo (B) inside the do loop. However checkpointing is done only after the optimum checkpointing time derived earlier as 613.188 minutes. Hence checkpointing is done 49 times. Also to recover the state required another 6 minutes plus another 306.594 minutes to do the recomputation plus 49 x 6 = 294 minutes to save the checkpoints.

Therefore the worst-case execution time of the program is 30,606.594 minutes.

e) If forked checkpointing is used, the saving of checkpoint is transparent as a child process performs it. Hence the only overhead is in restoring the data, which takes 6 minutes plus another 306.594 minutes to do the recomputation

Therefore the worst-case execution time is 30,312.594 minutes.

Problem 2.9:

Comparing the latest Top-500 list with the Top-500 Green list of HPC system, there are a lot differences. For example, the top 5 systems in the first list are:

1. Tianhe-1A at China's Nat's Supercomputer Center
2. Jaguar at Oak Ridge Nat'l Lab in USA
3. Nabelae at China's Nat's Supercomputer Center
4. TSUBAME 2.0 at GSIC Center, Tokyo Institute of Technology
5. Hopper at DOE/SC/ LBNL/NERSC

However, their rankings in the Green list are 11, 88, 14, 3 and 31 respectively. This indicates that high performance systems do not necessarily lead to high energy efficient systems. By looking at the above five systems' ranking in Green list, the TSUBAME 2.0 is clearly the winner, with China's two HPC systems also have good energy efficiency. On the other hand, Cray's two HPC systems have relatively low rankings in this list and are kind of losers in this sense.

There might be a reason behind this fact. If we look at the Tianhe-1A, Nabelae and TSUBAME 2.0, they all use the CPU + GPU architecture. This could be the major reason why their energy consumption is far less than the Cray's XT5 and XE6, which both use the CPU architecture only. Some estimation about the top 1 Tianhe-1A system is that, its power consumption could be more than 12 megawatts if it had been built only with CPUs. Another reason why TSUBAME 2.0 succeeded in green IT is its emphasis of energy consumption at the very begin. However, TSUBAME 2.0 has currently only 73,278 nodes while China' Tianhe-1A has 186,368 nodes.

Problem 2.10:

a) K computer

- It uses 88,128 2.0GHz 8-core SPARC64 VIIIfx processors
- It has a theoretical peak performance of 10 Pflops.
- It has a sustained speed of 8.162 Pflops according to Linpack benchmark.
- Its efficiency is 93%

Tianhe -1A

- It is equipped with 14,336 Xeon X5670 processors and 7,168 Nvidia Tesla M2050 general purpose GPU's. Hence it has a hybrid architecture.
- It has a theoretical peak performance of 4.701 Pflops.
- It has a sustained speed of 2.57 Pflops according to Linpack benchmark.
- Its efficiency is 54.6%

Jaguar

- It is equipped with Cray XT5-HE: MPP with 224,162 x86 AMD Opteron.
- It has a theoretical peak performance of 4.7 Pflops.
- It has a sustained speed of 1.76 Pflops according to Linpack benchmark.
- Its efficiency is 75.6%

b) K computer

- It uses a 6D Torus Interconnect called Tofu.
- High speed, highly scalable, high operability and high availability interconnect.
- Bidirectional bandwidth is 5GB/s
- Switch less implementation.

Tianhe -1A

- It has a custom designed ARCH Fat – Tree Interconnect.
- It is built with the InfiniBand DDR 4X and 98 TB of memory.
- The bidirectional bandwidth is 160Gbps.
- It has a latency of 1.57 microseconds for a node hop.
- Its aggregated bandwidth is 61 TB/second.
- At the first stage, 16 nodes are interconnected by a 16 port switching board, at the second stage all parts are connected to 11 384-port switches.

Jaguar

- It has a 3D Torus Interconnect.
- It uses a high-bandwidth, low-latency interconnect using the Cray SeaStar2+ router chips.
- The peak bidirectional bandwidth of each link is 9.6 GB/second.
- It directly interconnects all the nodes in the system avoiding the cost and complexity of external switches and allowing easy expansion.

Problem 2.11:

The authors build the SIOS by designing a special device driver for use in local disks. The drivers cooperate with each other to form the single address space. They work collectively to

maintain the data consistency among distributed data blocks. The file system running on in each node has the illusion that there is only one large virtual disk shared by all clients in the cluster.

The authors choose to design the RAID-x with SIOS support at the Linux driver level. This design provides a SSI to the users, demanding no file system modification. The SIOS is supported by a set of CDDs at the kernel level. There is no need to use a central server in our design. Each CDD maintains a peer-to-peer relationship with other CDDs.

Local disk access is faster than remote disk access. It is important to reduce the latency of remote disk access. The authors have developed a cooperative disk driver (CDD) which can achieves this goal. First, CDD preserves the RAID-0 bandwidth, while adding the fault tolerance capability. Second, it creates a SIOS across all distributed disks. Third, the CDD can implement other RAID configurations. Last, data consistency is maintained by the CDD instead of the file system. However, to scale up to a much larger number of disks, the communication protocol used in the CDD may become a performance bottleneck.

The authors have implemented in RAID-x a new disk mirroring mechanism for fault-tolerant. The RAID-x can tolerate all single-disk failures, same as that of RAID-5. Thus the RAID-x matches RAID-5 in terms of fault tolerance. Among the two architectures, the RAID-5 achieves single-fault tolerance with a much lower cost for using a fewer redundant disk space. However, the advantage of RAID-5 lies in its much improved erite performance, which is attributed to the property of orthogonality.

The RAID-x architecture demonstrates its strength in cluster I/O performance.

(a). The new RAID-x shows its strength in building distributed storage for serverless clusters. The orthogonal architecture is unique with orthogonal striping and mirroring. In the reliability area, the RAID-x can tolerate all single disk failures like the RAID-5. The background image writing also contributes the performance gain, especially in large write operations. The RAID-x completely eliminates the small-write problem by using no parity checks.

(b). The authors have developed new disk drivers at the Linux kernel level to support the SIOS in any ds-RAID configuration. These drivers enable faster remote disk access and parallel I/O without using a central file server. Much of the software overhead comes from heavy system calls in cluster I/O operations. The use of the CDDs with middleware reduces some overheads significantly. Benchmark performance results show scalable performance of RAID-x in cluster I/O operations.

(c). The Andrew benchmark shows the scalable advantage of using any ds-RAID over the use of a central NFS for cluster I/O operations. For parallel reads with 16 active clients, the RAID-x achieved a 1.5 times higher performance than other three RAID architectures. For parallel writes, RAID-x shows up to 2.2 times higher performance. The Bonnie benchmark shows that the RAID-x cuts 13% of the overhead in many I/O-centric operations.

(d). The authors have extended the Linux kernel to support the SIOS. This helps also implement the shared virtual memory and global file management. Many I/O-centric applications can benefit

by achieving scalable performance on a serverless cluster of computers. In particular, this will benefit data mining, transaction processing, and pervasive computing applications.

There are a few shortcomings of RAID-x:

(a) To scale up to a much larger number of disks, the communication protocol used in the CDD may become a performance bottleneck.

(b) A new distributed file system is desired to improve the efficiency of the disk drivers in using even faster networks. A more efficient data consistency protocol will provide higher scalability to hundreds or even thousands of disks in a very large ds-RAID system.

(c) The authors did not apply any data pre-fetching, or cooperative caching technique.

Problem 2.12:

(a). Routers and Switches:

IBM Blue Gene/L: The Blue Gene cluster was designed to achieve scalable performance, reliability through built-in testability, resilience by preserving locality of failures and checking mechanisms, and serviceability through partitioning and isolation of fault locations.

IBM Roadrunner: The Cell/B.E. processors provide extraordinary compute power that can be harnessed from a single multicore chip. The Cell/B.E. architecture supports a very broad range of applications. The first implementation is a single-chip multiprocessor with nine processor elements operating on a shared memory model. The rack is built with TriBlade servers, which are connected by an InfiniBand network.

In order to sustain this compute power, the connectivity within each node consists of four PCI Express x8 links, each capable of 2 GB/s transfer rates, with a 2 μs latency. The expansion slot also contains the InfiniBand interconnect, which allows communications to the rest of the cluster. The capability of the InfiniBand interconnect is rated at 2 GB/s with a 2 μs latency.

The Roadrunner uses MPI APIs to communicate with the other Opteron processors the application is running on in a typical single-program, multiple-data (SPMD) fashion. The number of compute nodes used to run the application is determined at program launch. The MPI implementation of Roadrunner is based on the open source Open MPI Project, and therefore is standard MPI. In this regard, Roadrunner applications are similar to other typical MPI applications such as those that run on the IBM Blue Gene solution. Where Roadrunner differs in the sphere of application architecture is how its Cell/B.E. accelerators are employed. At any point in the application flow, the MPI application running on each Opteron can offload computationally complex logic to its subordinate Cell/ B.E. processor.

Gray XT5: The Cray XT5 family employs an energy-efficient packaging technology, which reduces power use and thus lowers maintenance costs. The system's compute blades are packaged with only the necessary components for building an MPP with processors, memory, and interconnect. In a Cray XT5 cabinet, vertical cooling takes cold air straight from its source—the floor—and efficiently cools the processors on the blades, which are uniquely

positioned for optimal airflow. Each processor also has a custom-designed heat sink depending on its position within the cabinet.

Each Cray XT5 system cabinet is cooled with a single, high-efficiency ducted turbine fan. It takes 400/480VAC directly from the power grid without transformer and PDU loss. The Cray XT5 3D torus architecture is designed for superior MPI performance in HPC applications. This is accomplished by incorporating dedicated compute nodes and service nodes. Compute nodes are designed to run MPI tasks efficiently and reliably to completion.

Each compute node is composed of one or two AMD Opteron microprocessors (dual or quad core) and direct attached memory, coupled with a dedicated communications resource. Service nodes are designed to provide system and I/O connectivity and also serve as login nodes from which jobs are compiled and launched. The I/O bandwidth of each compute node is designed for 25.6 GB/second performance.

(b）Other technological comparisons:

IBM Blue Gene/L: With modular packaging, the Blue Gene/L system us constructed hierarchically from processor chips to 64 physical racks. This system was built with a total of 65,536 nodes with two PowerPC 449 FP2 processors per node. The 64 racks are interconnected by a huge 3D 64 x 32 x 32 torus network.

IBM Roadrunner: The system was used mainly to assess the decay of the U.S. nuclear arsenal. The system has a hybrid design with 12,960 IBM 3.2 GHz PowerXcell 8i CPUs (Figure 2.26) and 6,480 AMD 1.8 GHz Opteron 2210 dual-core processors. In total, the system has 122,400 cores. Roadrunner is an Opteron cluster accelerated by IBM Cell processors with eight floating-point cores.

The InfiniBand switches cluster together 18 connected units in 270 racks. In total, the cluster connects 12,960 IBM Power XCell 8i processors and 6,480 Opteron 2210 processors together with a total of 103.6 TB of RAM. This cluster complex delivers approximately 1.3 Pflops. In addition, the system's 18 Com/ Service nodes deliver 4.5 Tflops using 18 InfiniBand switches.

The second storage units are connected with eight InfiniBand switches. In total, 296 racks are installed in the system. The tiered architecture is constructed in two levels. The system consumes 2.35 MW power, and was the fourth most energy-efficient supercomputer built in 2009.

Gray XT5: The basic building blocks are the compute blades. The interconnect router in the SeaStar+ chip provides six high-speed links to six neighbors in the 3D torus. The system is scalable by design from small to large configurations. The entire system has 129 TB of compute memory. In theory, the system was designed with a peak speed of Rpeak = 2.331 Pflops. In other words, only 75 percent (=1.759/2.331) efficiency was achieved in Linpack experiments. The external I/O interface uses 10 Gbps Ethernet and InfiniBand links. MPI 2.1 was applied in messagepassing programming.

The system consumes 32–43 KW per cabinet. With 160 cabinets, the entire system

consumes up to 6.950 MW. The system is cooled with forced cool air, which consumes a lot of electricity. The Cray XT5 system incorporates a high-bandwidth, low-latency interconnect using the Cray SeaStar2+ router chips. The system is configured with XT5 compute blades with eight sockets supporting dual or quad-core Opterons.

The XT5 applies a 3D torus network topology. This SeaStar2+ chip provides six high-speed network links which connect to six neighbors in the 3D torus. The peak bidirectional bandwidth of each link is 9.6 GB/second with sustained bandwidth in excess of 6 GB/second. Each port is configured with an independent router table, ensuring contention-free access for packets.

The router is designed with a reliable link-level protocol with error correction and retransmission, ensuring that message-passing traffic reliably reaches its destination without the costly timeout and retry mechanism used in typical clusters. The torus interconnect directly connects all the nodes in the Cray XT5 system, eliminating the cost and complexity of external switches and allowing for easy expandability. This allows systems to economically scale to tens of thousands of nodes — well beyond the capacity of fat-tree switches. The interconnect carries all message-passing and I/O traffic to the global file system.

Problem 2.13:

a. Take SGI UV for an example, SGI UV Models as follows:

SGI UV 10: In a 4U rackmount form factor, SGI UV 10 packs up to four sockets (40 cores, 80 threads), 64 DIMM slots and ten I/O expansion slots.

SGI UV 100: Based on an industry-standard 19" (48.26cm) rackmount 3U form factor, SGI UV 100 addresses the mid-range market. SGI UV 100 scales to 96 sockets (960 cores, 1920 threads) and 12TB of shared memory in two racks.

SGI UV 1000: For maximum scalability, SGI UV 1000 ships as a fully integrated cabinet-level solution with up to 256 sockets (2560 cores, 4096 threads) and 16TB of shared memory in four racks.

The SGI UV is a scalable global shared memory system based on the SGI NUMAflex architecture. Physically, these systems are similar to the SGI Altix? 4700 and SGI Altix 450, with a compact blade design, a NUMAflex architecture that supports a large number of processors in a global shared memory configuration running a single copy of standard Linux and the ability to share memory across multiple system images across SGI NUMAlink connections.

The SGI UV is designed to extend the industry leading scalability of SGI shared memory systems in all dimensions — processors, memory, interconnect and I/O. The current release supports up to 2,560 cores and 16TB of memory per Single System Image (SSI). Much larger multi-partition systems are supported with shared global memory address out to multiple petabytes, taking the company's leadership in shared memory systems to new heights. Configuration options enable the creation of systems that are optimized for compute capability (maximum core count), maximum total memory, system bisection bandwidth or maximum I/O.

SGI UV distinguishes itself from standard cluster approaches by tightly integrating with the Intel? Quick Path Interconnect (QPI). This integration provides both global shared memory and efficient access to that memory by working at the cache line level over the entire system consisting of tens to tens of thousands of blades. This cache-line-oriented approach contrasts sharply with cluster based approaches which are optimized for transferring large amounts of data over an InfiniBand interconnect which is connected to an I/O channel.

The SGI UV architecture also integrates multiple computing paradigms into a single environment based on industry standard Intel? Xeon processors. The design increases the efficiency Intel? Xeon scalar CPUs by providing vector memory operations, a rich set of atomic memory operations, and tight integration of application-specific hardware such as GPUs, digital signal processors and programmable hardware accelerators.

b. With up to 1TB of shared memory, SGI UV 10 delivers up to .289 teraflops of compute power in a single system. SGI UV 10 is ideal for a diverse set of technical computing and enterprise applications. SGI UV 100 delivers up to 9.2 teraflops of compute power in a single system image. SGI UV 1000 delivers up to 24.6 teraflops of compute power in a single system image.

c. SGI UV 1000 includes: 42U high rack; 16 blades maximum per enclosure; 2 enclosures maximum per rack; 64 sockets (640 cores, 1280 threads) maximum per rack; 4 rack maximum SSI size; Maximum 256 sockets (2560 cores, 4096 threads) per 4 rack system; Maximum 8 terabytes per rack (16TB total per single system); Architectural provisioning for up to 32,000 blades.

The features of SGI UV as follows: Modular Blade Design -- Multiple options in compute blades with respect to number of cores, processor speed and memory. Compatible with Future Technology -- Socket compatible with future Intel Xeon processors. Scalable System Size -- Scales to 256 sockets (2560 cores, 4096 threads) system size and as much as 16TB memory.

NUMAlink 5 (NL5) Interconnect, MPI Support -- High bandwidth, very low latency, MPI Offload Engine. SGI Reliability, Availability and Serviceability (RAS) -- Fault prevention, detection, recovery and memory RAS. Standards-Based Design -- Industry standard CPUs, memory and I/O. Certified on Multiple Operating System -- Industry-standard Linux (SUSE or RedHat) and Microsoft Windows Server 2008 R2.

Problem 2.14:

(a). (1) Availability of a server = MTTF / (MTTF + MTTR)
In this case, Mean time to failure (MTTF) = 200 days
Mean time to repair (MTTR) = 5 days
Hence availability is 200 / 205 = 0.9756 = 97.56%
The server is unavailable for time = 2.44%. Furthermore, due to the scheduled maintenance, the server is unavailable for 1 day every week i.e. 52/365 = 14.2%. Both servers will be down for time = 0.059%

Downtime due to maintenance = 14.2%

Therefore the total time the servers are unavailable is 14.259%

Hence the availability of the 2 servers is **85.741%**

If we ignore the fact that a server can be down when the other is being maintained the availability will be **99.941**%

(2) The single points of failure are the SCSI bus and the disk. Because if these fail, the servers cannot operate.

(b). The availability of the disk is given by 800 / (800 + 20) = 97.5%

Therefore total unavailability time is 2.5% for the disk.

Similarly the total unavailable time for the SCSI bus is 2%

Therefore the total downtime of the cluster is

0.059% + 14.2% + 2.5% + 2% = 18.759%

Hence the total availability of the cluster is **81.241%**

If we ignore the fact that a server can be down when the other is being maintained the availability will be **95.441**%

(c) To remove the SCSI single point of failure we will have 2 SCSI buses running on both sides of the servers so that if one fails it can transmit through the other one. To remove the single point of failure due to the disk array we can use a RAID 1 configuration in which two copies of the data are kept on two disks so that if one fails the other can take over.

Problem 2.15

(a) Non-preemptive vs. preemptive: In non-preemptive scheduling, once a process has been given the resource, the resource cannot be taken away from that process. Therefore, the advantages are that the response times are more predictable and there is no overhead to suspend and resume the process. However, a later high-priority process may be delayed. In preemptive scheduling, this problem could be avoided, but it has large overhead and implementation complexity.

Suggestion: we could set a priority threshold. When the priority of the incoming process is less the threshold, non-preemptive scheduling is used. It is the case for most of time. However, when some really critical process comes with higher priority than the threshold, preemptive one is used to reduce waiting time.

(b) Static vs. dynamic scheduling: Static scheduling is simple, but may underutilize the cluster resource and cannot handle node failure. Dynamic scheduling improves the system utilization but is difficult to implement, requiring cooperation between a running job and the JMS.

Suggestion: we may develop some light-weighted dynamic scheduling mechanisms to reduce the implement overhead.

(c) Dedicated vs. space-sharing: Dedicated scheduling has poor system utilization while space-sharing scheduling has large-job problem.

Suggestion: develop better algorithm to alleviate the large-job problem; restrict the types of jobs that can be run in dedicated mode.

(d) Explain time-sharing, Independent vs. gang scheduling:

In time-sharing mode, multiple user processes are assigned to the same node. Independent scheduling is to use the operating system of each node to schedule different processes. Gang scheduling scheme schedules all processes of a parallel job together. The problem of independent scheduling is severe slowdown due to inefficient interaction among nodes. However, the gang scheduling has the skew problem beside implementation complexity.

Suggestion: we may provide some coordination mechanisms on top of the independent scheduling to achieve better performance.

(e) Difference in stay and migrating polices

In the stay policy, the cluster process stays in the local node to compete with local processes, which slows down both local and cluster processes. In migrating policy, the jobs can be flowed around other available nodes for better load balancing. However, it has implementation overhead.

Problem 2.16:

Examples and implementation obstacles are listed below:

(a) Single entry point: (GLUnix)

Since system needs to transparently distributes the user's login connection requests to different physical hosts, there are several implementation obstacles such as where to put the home directory as well as authentication and host failure problems.

(b) Single memory space: (TreadMarks)

As single memory space creates an illusion of centralized main memory while the memory are physically distributed, it might need to take care of the non-uniform memory access time, which requires software and hardware support.

(c) Single file hierarchy: (GLUnix, TreadMarks, DQS, LSF)

Stable storage requires both local and remote storages. Either centralized or distributed implementation has severe problems as mentioned in the chapter, such as single point of failure for the former

(d) Single I/O space: (Some RAID systems)

Single I/O space requires a distributed disk array to appear as a single space. This poses great challenges on the reliability and security issues, as well as consistency problem.

(e) Single network space: (None)

A single network space may implement some transparent master/slave replication scheme in order to support regions of synchrony, which requires significant implementation efforts.

(f) Single networking: (None)

As any process on any node can use any network, there must be some standards or conventions that support the communication of network packets of different formats, which is

very challenging.

(g) Single point of control: (GLUnix, DQS, LSF)

The system administrator should be able to configure, monitor and control the entire cluster from a single point, which requires additional software or hardware support. For example, a system console that is connected to all nodes of the cluster may be needed.

(h) Single job management:(LSF)

In order to do single job management, one requirement is the need of job status information across the cluster. Moreover, this information should be acquired as soon as possible to avoid it from becoming stale. This requires a latency-aware high performance interconnection network.

(i) Single user interface: (LSF)

In this case, we need to provide single graphical interface. Some efforts should be made in developing a cluster GUI through the use of web technology.

(j) Single process space: (TreadMarks)

As all user processes belong to a single process space and share process identification scheme, a cluster-level process id management may be needed. This requires software, hardware or middle ware support.

Problem 2.17:

a. Serial jobs run on a single node. Parallel jobs use multiple nodes.

b. Interactive jobs are those that require fast turnaround time, and their input/output is directed to a terminal. These jobs do not need large resources, and users expect them to execute immediately, not to wait in a queue. Batch jobs normally need more resources, such as large memory space and long CPU time. But they do not need immediate responses. They are submitted to a job queue to be scheduled to run when the resource becomes available (e.g., during off hours).

c. Foreign jobs are created outside the JMS. For instance, when a network of workstations is used as a cluster, users can submit interactive or batch jobs to the JMS. Meanwhile, the owner of a workstation can start a foreign job at any time, which is not submitted through the JMS. Such a job is also called a local job, as opposed to cluster jobs (interactive or batch, parallel or serial) that are submitted through the JMS of the cluster. The characteristic of a local job is fast response time. The owner wants all resources to execute his job, as though the cluster jobs did not exist.

d. The workstation's cycles can be divided into three portions, for kernel processes, local processes, and cluster processes. However, to stay slows down both the local and the cluster jobs, especially when the cluster job is a load balanced parallel job that needs frequent synchronization and communication. This leads to the migration approach to flow the jobs around available nodes, mainly for balancing the workload.

e. Three schemes are used to share cluster nodes. In the dedicated mode, only one job runs in the cluster at a time, and at most, one process of the job is assigned to a node at a time. The

single job runs until completion before it releases the cluster to run other jobs. Note that even in the dedicated mode, some nodes may be reserved for system use and not be open to the user job. Other than that, all cluster resources are devoted to run a single job. This may lead to poor system utilization.

The job resource requirement can be static or dynamic. A common scheme is to assign higher priorities to short, interactive jobs in daytime and during evening hours using tiling. In this space-sharing mode, multiple jobs can run on disjointed partitions (groups) of nodes simultaneously. At most, one process is assigned to a node at a time. Although a partition of nodes is dedicated to a job, the interconnect and the I/O subsystem may be shared by all jobs. Space sharing must solve the tiling problem and the large-job problem. In space-sharing model, only one user process is allocated to a node. However, the system processes or daemons are still running on the same node. In the time-sharing mode, multiple user processes are assigned to the same node.

f. The most straightforward implementation of time sharing is to use the operating system of each cluster node to schedule different processes as in a traditional workstation. This is called local scheduling or independent scheduling. However, the performance of parallel jobs could be significantly degraded. Processes of a parallel job need to interact. For instance, when one process wants to barrier-synchronize with another, the latter may be scheduled out. So the first process has to wait. As the second process is rescheduled, the first process may be swapped out.

The gang scheduling scheme schedules all processes of a parallel job together. When one process is active, all processes are active. The cluster nodes are not perfectly clock-synchronized. In fact, most clusters are asynchronous systems, and are not driven by the same clock. Although all processes are scheduled to run at the same time, they do not start exactly at the same time. Gang-scheduling skew is the maximum difference between the time the first process starts and the time the last process starts. The execution time of a parallel job increases as the gang-scheduling skew becomes larger, leading to longer execution time. In a homogeneous cluster, gang scheduling is more effective. However, it is not yet realized in most clusters, because of implementation difficulties.

Problem 2.18:

(a). Serial jobs run on a single node. Parallel jobs use multiple nodes. Interactive jobs are those that require fast turnaround time, and their input/output is directed to a terminal. These jobs do not need large resources, and users expect them to execute immediately, not to wait in a queue. Batch jobs normally need more resources, such as large memory space and long CPU time. But they do not need immediate responses. They are submitted to a job queue to be scheduled to run when the resource becomes available.

(b). The entire cluster has one master LIM. In this mode, the load of one master LIM is usually too large, and the master LIM is the single point of failure in entire cluster. When all LIMs are

masters, it needs to exchange information between each master, if all LIMs are masters, the information load will be large, and the bandwidth consumption also huge.

(c). In the LSF master-election scheme, it needs to collect information of all the nodes, and then elects a new master, so the wait time increases with the node number.

Problem 2.19:

MOSIX : Advantages -

- It allows applications to run in remote nodes as if they are run locally.
- Users can login on any node and do not need to know where their programs run.
- No need to modify or link applications with special libraries.
- No need to copy files to remote nodes.
- Load-balancing.
- Migrating processes from slower to faster nodes and from nodes that run out of free memory.
- Secure run time environment (sandbox) for guest processes.
- Checkpoint and recovery.

Shortcomings -

- There is no Job Scheduler module.
- No possibility of defining resource usage policies in a cluster.
- Need to establish trust among the node owners, as they are not secured.

Problem 2.20:

Architecture design:

Tianhe-1A system consists of five major components. The compute subsystem houses all the CPUs and GPUs on 7,168 compute nodes. The service subsystem comprises eight operation nodes. The storage subsystem has a large number of shared disks. The monitoring and diagnosis subsystem is used for control and I/O operations. The communication subsystem is composed of switches for connecting to all functional subsystems.

The basic building blocks of Cray Jaguar are the compute blades. The interconnect router in the SeaStar+ chip provides six high-speed links to six neighbors in the 3D torus. The system is scalable by design from small to large configurations. The entire system has 129 TB of computing memory. The external I/O interface uses 10 Gbps Ethernet and InfiniBand links. MPI 2.1 was applied in message passing programming.

Resource manager:

The total disk storage of Tianhe-1A is 2 Petabytes implemented as a Lustre clustered file system, and the total memory size of the system is 262 Terabytes. Another significant reason for the increased performance of the upgraded Tianhe-1A system is the Chinese-designed NUDT custom designed proprietary high-speed interconnect called Arch that runs at 160 Gbps, twice

the bandwidth of InfiniBand. Jaguar uses an external Lustre file system called Spider for all file storage. The file system read/write benchmark is 240 GB/s, and provides over 10 petabytes (PB) of storage.

Software environment:

The software stack on the Tianhe-1A is typical of any high-performance system. It uses Kylin Linux, an operating system developed by NUDT and successfully approved by China's 863 Hi-tech Research and Development Program office in 2006. Kylin is based on Mach and FreeBSD, is compatible with other mainstream operating systems, and supports multiple microprocessors and computers of different structures. Kylin packages include standard open source and public packages, which have been brought onto one system for easy installation.

The system features FORTRAN, C, C++, and Java compilers from Intel (icc 11.1), CUDA, OpenMP, and MPI based on MPICH2 with custom GLEX (Galaxy Express) Channel support. The NUDT builder developed a mathematics library, which is based on Intel's MKL 10.3.1.048 and BLAS for the GPU based on NVIDIA and optimized by NUDT. In addition, a High Productive Parallel Running Environment (HPPRE) was installed. This provides a parallel toolkit based on Eclipse, which is intended to integrate all the tools for editing, debugging, and performance analysis. In addition, the designers provide workflow support for Quality of Service (QoS) negotiations and resource reservations.

The Cray Jaguar family run the Cray Linux Environment, formerly known as UNICOS/lc. This incorporates SUSE Linux Enterprise Server and Cray's Compute Node Linux.

Reported applications:

Most of applications of Tianhe-1A are specially tailored to satisfy China's national needs. (Parallel AMR method; Parallel eigenvalue problems; Parallel fast multipole methods; Parallel computing models; Gridmol computational chemistry; ScGrid middleware, grid portal; PSEPS parallel symmetric eigenvalue package solvers; FMM-radar fast multipole methods on radar cross sections; Transplant many open source software programs; Sandstorm prediction, climate modeling, EM scattering, or cosmology; CAD/CAE for automotive industry)

Hundreds of applications have been ported to run on the Cray XT series, many of which have been scaled up to run on 20,000 to 150,000 processor cores. The petaflop Jaguar seeks to address some of the most challenging scientific problems in areas such as climate modeling, renewable energy, materials science, seismology, chemistry, astrophysics, fusion, and combustion. Annually, 80% of Jaguar's resources are allocated through DOE's Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program, a competitively-selected, peer-reviewed process open to researchers from universities, industry, government, and non-profit organizations.