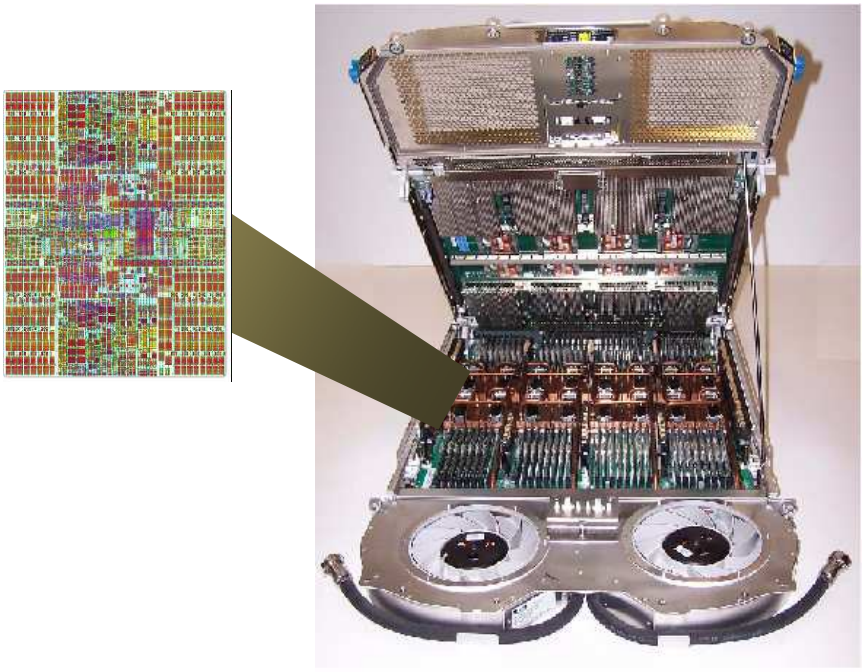
IBM Tuning Guide
For High Performance Computing Applications
On IBM POWER6



Luigi Brochard, Louis Capps, Don DeSota , Jim Edwards, Brad Elkin, Lawrence Hannon , John Lewars, Jim McInnes, Frank O'Connell, Raj Panda, Mathias Puetz, Joe Robichaux and Steve White

Release 1.0
April 7, 2009
IBM Systems and Technology Group

Contents

Figures

Tables

1 Introduction

In June of 2007, IBM introduced the IBM Power 570[1], the first system using the new POWER6 microprocessor. Based upon proven Simultaneous Multi-threading (SMT) and dual core technology from POWER5, POWER6 extended IBM's leadership position by coupling high frequency core design with a cache hierarchy and memory subsystem specifically tuned for ultra high frequency multi-threaded cores.

Following the success of the Power 570, in early 2008, IBM announced the IBM Power 575, the first POWER6 system specifically designed for High Performance Computing (HPC) and the follow-on product to the POWER5 based System p 575 or p5-575. The Power 575 shipped with full support of AIX and Linux, updates to IBM's HPC stack (Parallel Environment, compilers, HPC toolkit, etc.) and a tuned Infiniband fabric using custom Infiniband HCAs designed to support thousands of clustered CPU cores.

It is the goal of this paper to explore in detail performance aspects of HPC type workloads on POWER6 and compare with previous POWER generations. Much of the work and knowledge in this paper was derived from running workloads on the IBM Power 570 midrange server system as well as IBM Power 575 cluster systems. While the Power 570 is a POWER6 16 core system based on 4-core scalable building blocks targeted mainly for high performance database and other commercial applications, because of its extremely well balanced design, the Power 570 also performs well with small HPC workloads and is an excellent platform for this performance study.

To help the intended technical readers of this paper, such as HPC application specialists, we've included suitable examples to articulate the quantitative performance differences between POWER5+ and POWER6. In addition, we've used a "FAQ" approach to answer questions on POWER6 performance. Some of the typical questions that HPC specialists ask are,

1. How do you get the best performance on POWER6?
2. How does SMT benefit performance on POWER6?
3. What is the performance of legacy executables on POWER6?
4. How to optimize MPI application performance on a Power 575 cluster?

For a more in depth description of the POWER6 architecture, please refer to the IBM Journal of R&D, Volume 51, November 6, 2007.[2]

[1] See http://www.ibm.com/systems/p/hardware/midrange/570m/index.html) for more information.
[2] http://www.research.ibm.com/journal/rd51-6.html

2 POWER6 Microarchitecture Overview

POWER6 represents significant innovation in multiprocessor architecture driving ground breaking logic and circuit design. While most vendors have stalled in frequency improvements, POWER6 proves that a significant performance boost along with frequency can be implemented by focusing on simpler logic and gate level power management while maintaining per clock performance. In addition to its high frequency core, larger private L2 caches, multiple on-chip memory controllers and enhanced SMP interconnect provide bandwidth to keep the cores operating at peak capacity. The large, victim L3 cache, similar in design to previous generations is shared by both cores on the chip, and accessed in parallel with the L2 caches. Other key innovations in the SMP interconnect fabric and associated logical system topology, enable improved RAS, virtualization, and dynamic configuration capabilities.

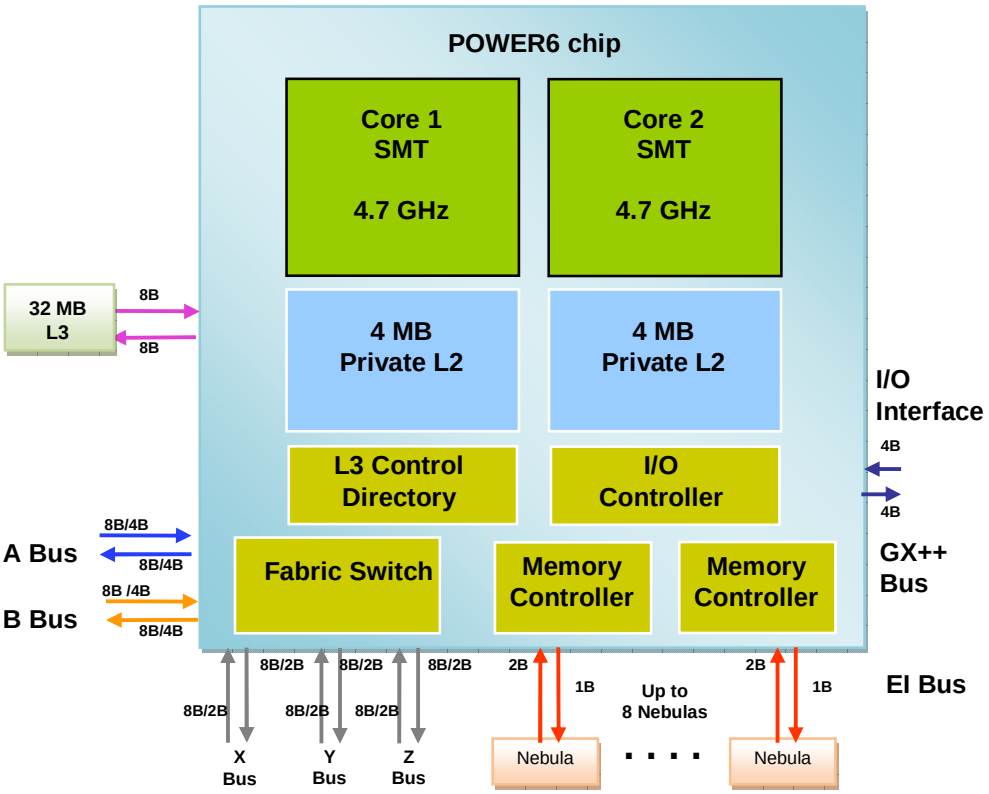
The POWER6 processor[3] in Figure 2-1 is based on 64-bit PowerPC architecture and incorporates[4]:

- Two ultra-high frequency processor cores with two SMT (Simultaneous Multithreading) threads per processor (four threads per chip)
- One 128 bit VMX/AltiVec (Vector Multimedia Extension) unit per processor
- Private 4 MB L2 cache for each processor core
- Up to 32 MB L3 cache controller shared by the two processor cores
- Up to two integrated memory controllers
- Integrated I/O controller
- Integrated SMP coherence and data interconnect switch that enables scalable connectivity for up to 32 POWER6 chips to enable a 64-way SMP
- Support logic for dynamic power management, dynamic configuration and recovery, and system monitoring.

[3] Different versions of POWER6 processor are used in different POWER6 server systems. In general, POWER6 processors differ from one another in terms of core frequency, number of memory controllers, L3 cache size and fabric bus characteristics. See IBM System Power 575 Compute Node.

[4] See IBM Journal of R&D, Volume 51, November 6, 2007. for more detail. (http://www.research.ibm.com/journal/rd51-6.html)

Figure 2-1. POWER6 Chip



A primary design goal for POWER6 was to reach high operating frequencies and performance while maintaining reasonable power dissipation. The high level design to achieve these goals required simplification of the functionality of the CPU core. Also, in contrast to POWER5+, much of POWER6 was designed with in-order instruction execution. To improve CPI and ensure efficient use of this design point:

- Changes were made in the areas of prefetch, predecode, and L1 cache design to keep the pipelines busy in addition to changes in all areas of the core architecture including instruction dispatch, issue and grouping.
- Several compiler innovations were implemented to optimize use of the POWER6 design approach
- Significant enhancements to SMT (see 3.2 for more detail) such as simultaneous instruction fetch were made to ensure units inside POWER6 are kept fully utilized

Alternate approaches were used in POWER6 for the POWER5+ store forward and dedicated fixed-point multiply/divide functions. Additional innovation in cache and fabric design was used to handle physical constraints like wire delays and memory access times that do not scale with the core clock frequency from 1.9 GHz to 4.7 GHz.

The new approach for POWER6 provides some codes an amazing 2X speedup over POWER5+. And while not all codes scale directly with frequency from POWER5+, few new designs see the type of double digit gains in performance than have been achieved with POWER6. In addition, even with its in-order architecture and significantly different design, POWER6 is able to provide gains while using legacy codes and code techniques. In some cases where the customer is striving for maximum efficiency of their codes, an understanding of POWER6 architecture is important and application implementation should be approached from a different perspective than has been used in the past for traditional superscalar CPUs like POWER4 and POWER5.

3 Performance Optimization on POWER6

Applications built for a common PowerPC architecture or even specific POWER processors such as POWER4 or POWER5 will run on the POWER6 processor. However, the POWER6 architecture has several new features that differ from previous versions of the POWER chip. By utilizing the latest compilers and effectively exploiting simultaneous multi-threading (SMT) one can often obtain significant performance improvements.

3.1 Compilers

3.1.1 FORTRAN and C compiler release levels needed

IBM XL Fortran 11.1 and IBM XL C/C++ 9.0 compilers support use of the POWER6 specific tuning options, such as –qarch=pwr6 and –qtune=pwr6.

Certain versions of gcc 4.1 and gcc 4.2 support POWER6 specific optimization using –mcpu=power6.

The remainder of this document will focus on utilizing the IBM XL compilers.

3.1.2 IBM FORTRAN and C Compiler options

Detailed information about the XL FORTRAN 11.1 and C/C++ 9.0 compilers can be found here.[5]

For those familiar with the XL compilers, the following list of compiler options include a short summary of new options and those related to POWER6 optimizations.

3.1.3 General Tuning Options

`qtune=pwr6`

- Instructs the compiler to schedule instructions for POWER6 optimization. This can be used with different –qarch options, but most commonly used with –qarch=pwr6.

`qtune=balanced`

- When used with **-qarch=pwr5** (or pwr5x) this option will generate a binary that runs on both POWER5 and POWER6 systems, but with scheduling improvements that **should** improve POWER6 performance.

`qarch=pwr6`

- Utilize POWER6-specific instructions. Compiling with –qarch=pwr6 –qtune-pwr6 should yield optimal performance on the POWER6. Note compiling with –qarch=pwr6 will generate an executable that will only run on POWER6 or later processors.

`qfloat=norngchk`

- This new option produces faster software divide and square root sequences. Eliminates control flow in the software div/sqrt sequence by not checking for some boundary cases in input values. The optimization is used by default with –O3 unless –qstrict is also specified.

[5] http://publib.boulder.ibm.com/infocenter/comphelp/v9v111/index.jsp

3.1.4 Diagnostic Options

`qsmp=stackcheck`
- Detect a thread's stack going beyond its limit.

`qoptdebug`
- Improve debugging of optimized code.

3.2 SMT (Simultaneous Multi-threading)

SMT for POWER5 has provided significant benefit in many commercial workloads but not always for HPC workloads. The benefit using SMT on POWER6 is much higher even on HPC workloads. In fact, POWER6 SMT shows significant throughput improvements for most HPC loads or parallel applications and is highly recommended for achieving maximum throughput from each POWER6 core.

The following data were collected on two systems:

- A POWER5 p575+ system running at 1.9 GHz
- A Power 570 system with 4 cores running at 4.7 GHz

Figure 3-1 compares the performance for several benchmarks from the NAS Parallel Benchmark Class C suite running with and without SMT. Times reported are for:

- binaries tuned for POWER5 and data collected on a p575+ system (using the **p5-p5** legend)
- binaries tuned for POWER5 and data collected on a Power 570 system (using the **p5-p6** legend)
- binaries tuned for POWER6 and data collected on a Power 570 system (using the **p6-p6** legend)

Figure 3-1 Performance of NAS Parallel Benchmarks: SMT vs. ST

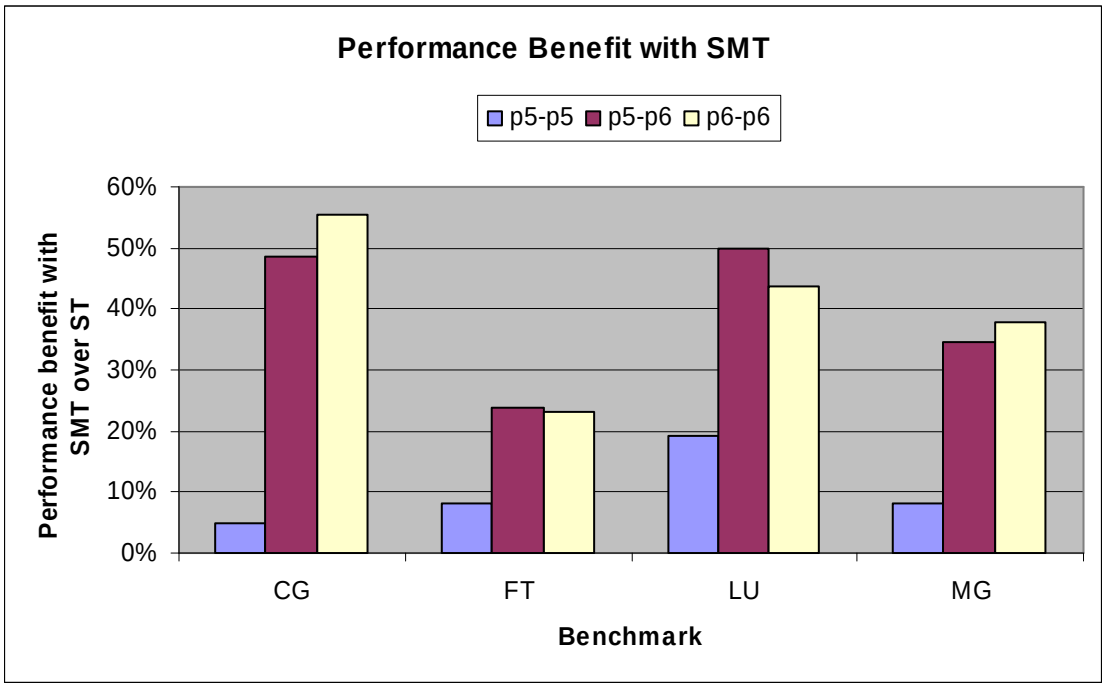


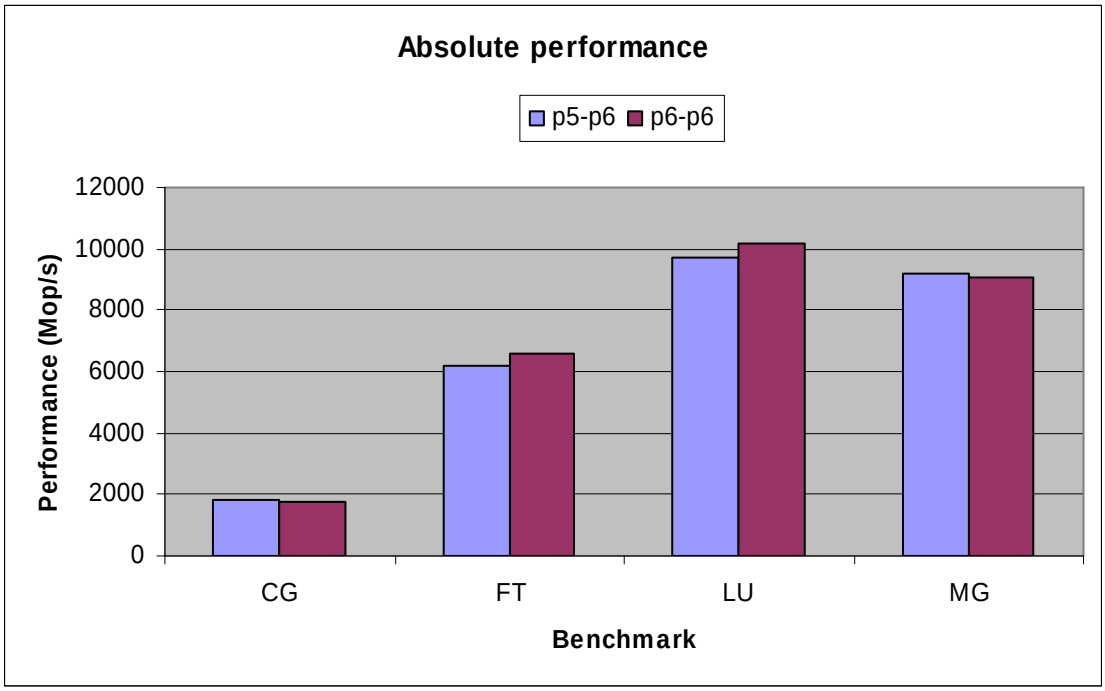
Figure 3-2 compares the performance of POWER5 and POWER6-tuned binaries from the NAS Parallel Benchmarks running 8 threads in SMT mode.

The times reported are for:

- binaries tuned for POWER5 and data collected on a Power 570 system (using the **p5-p6** legend)
- binaries tuned for POWER6 and data collected on a Power 570 system (using the **p6-p6** legend)

This figure shows that the POWER5-tuned and POWER6-tuned binaries have nearly equivalent performance when running in SMT mode.

Figure 3-2 Performance of NAS Parallel Benchmarks: SMT



3.3 Analysis

On POWER5+, you cannot dispatch the two SMT threads simultaneously; while on POWER6 you can. As shown in Figure 3-2, when run using SMT, binaries tuned for POWER5 and POWER6 perform comparably. A possible explanation for this is that SMT hardware takes advantage of any stalls or other bottlenecks encountered by the primary thread running on a POWER6 core to execute the secondary thread. When a core time for a thread is spent waiting on stalls, the SMT hardware can use the idle functional units to work on the other SMT thread. When the stalls are reduced by tuning for POWER6, SMT hardware doesn't have as much stall time to use, leaving the overall execution time about the same as before. The conclusion: When running in SMT mode, POWER6 performs well even on binaries which are not tuned for POWER6.

3.4 SMT on single node

SMT provides significant speedups for many HPC applications – Molecular Dynamics, CFD, Weather, Ocean Modeling, etc. In Table 3-1, we show the SMT benefit for a number of applications. All of these applications were compiled for POWER6 and run on a 4-core Power 570 4.7 GHz system in ST (4 tasks) and SMT (8 tasks) mode. For several of the applications, multiple benchmarks were run and an average SMT benefit over the set of benchmarks is shown in the table below.

Table 3-1 SMT benefit on HPC applications

Application	Area	Performance Ratio (SMT/ST)
AMBER[6]	Molecular Dynamics	56%
NAMD[7]	Molecular Dynamics	32%
FLUENT[8]	CFD	91%
STAR-CD[9]	CFD	62%
WRF	Weather	48%
POP	Ocean Modeling	46%

3.5 SMT with MPI

On POWER5+, for purely MPI applications with a large number of MPI tasks, SMT may not provide much benefit. In fact, the overall performance of a job could drop when run in SMT mode, i.e. by doubling the number of MPI tasks. A doubling of the task count increases overall communication time and when this is not offset by a reduction in compute time due to SMT, there will be a drop in overall performance.

On POWER6, SMT can provide a significant performance benefit as shown in Table 3-1 that can potentially offset an increase in communication time due to a doubling of the task count. In section 5.4.3, we show the SMT performance benefit on a suite of MPI application benchmarks. For hybrid (MPI+OpenMP) applications, a more effective way of using SMT on POWER6 can be to double the number of OpenMP threads instead of the number of MPI tasks.

[6] Average for 8 AMBER standard benchmarks
[7] Average for 2 benchmarks
[8] Average for 3 large FLUENT standard benchmarks
[9] Class A benchmark

4 Running Legacy Executables on POWER6

POWER6 is binary compatible with previous generations of PowerPC microprocessors. While significant gains have been seen running unmodified legacy PowerPC code on POWER6, even more performance can be realized by taking advantage of specific enhancements and architectural features of POWER6. In particular, the use of SMT has been shown to provide more benefit on POWER6.

4.1 Introduction

Expanding on the theme from Section 2 and architectural differences between POWER6 and previous generations, the recommended approach in ST mode is to create a binary targeted for the particular architecture where it can be run whenever possible. When running in SMT mode, the performance distinction between different binaries is much smaller.

Benchmark results from GAMESS[10] application (run with the crown-ether dataset) are used to illustrate typical application performance of POWER5 and POWER6 binaries on POWER5 and POWER6 systems. GAMESS is a well-known general ab initio quantum chemistry package.

4.2 Build/Execution Details

GAMESS builds were done with XL Fortran compiler version 11.1 and XL C/C++ compiler version 9.0 GA compilers.

Three sets of compile options were used:

```
-O3 –qarch=pwr5 –qtune=pwr5x              (set 1)
-O3 –qarch=pwr5 –qtune=balanced           (set 2)
-O3 –qarch=pwr6 –qtune=pwr6               (set 3)
```

The first two binaries were run on a (POWER5+) p575+ system running at 1.9 GHz. All three binaries were also run on a Power 570 system running at 4.7 GHz (the results for a Power 575 should show similar results). All runs were bound to 4 cores in either ST (4 tasks) or SMT (8 tasks) mode.

[10] http://www.msg.ameslab.gov/GAMESS/

4.3 Data

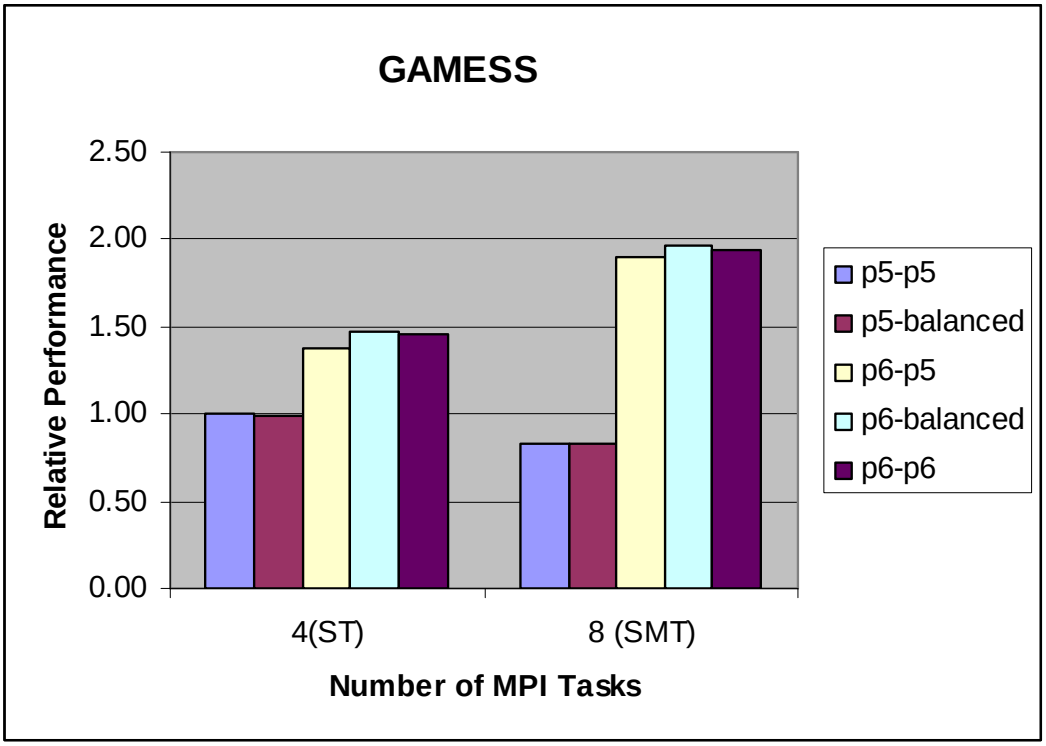
In Figure 4-1, the performance for each group is reported relative to the benchmark compiled with the options from set 1 in Section 4.2 and run on a POWER5+ p575+ system using 4 threads on 4 cores (referred to in the figure as **p5-p5**).

The legend notation corresponds with:

- **p5-p5** - a binary compiled with the set 1 options and run on a p575+
- **p5-balanced** - a binary compiled with the set 2 options and run on a p575+
- **p6-p5** - a binary compiled with the set 1 options and run on a Power 570,
- **p6-balanced** - a binary compiled with set 2 and run on a Power 570, and
- **p6-p6** - a binary compiled with set 3 and run on a Power 570.

Benchmark labels denote whether the benchmark is run with 4 tasks (1 task per core) or with 8 tasks (SMT mode; 2 tasks per core).

Figure 4-1 GAMESS runs with the crown ether dataset.



4.4 Discussion

- Note how running with 8 tasks (2 tasks per core- SMT mode) slows down execution on POWER5+ while speeding up execution on the POWER6.

- POWER5 binaries can perform 40-50% faster on POWER6 using 1 task per core. Using POWER6 SMT mode (2 tasks per core) can increase the margin even further, with POWER6 running nearly 100% faster than POWER5 ST or SMT mode.

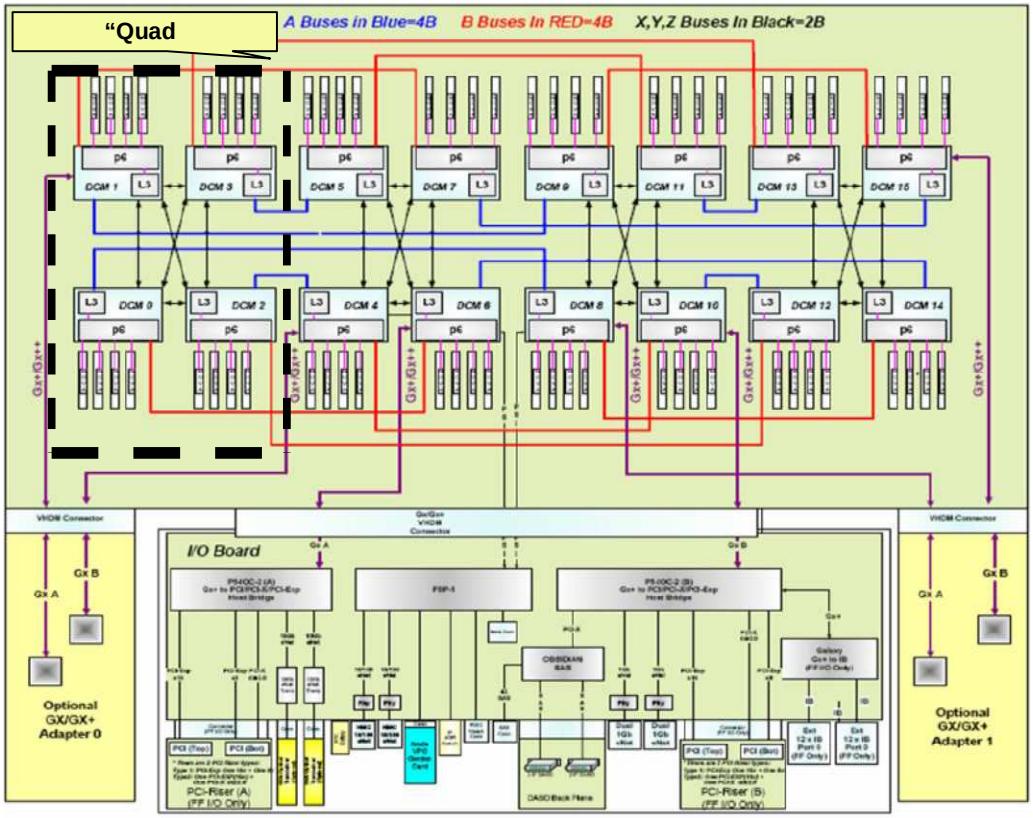
- At this time, POWER6-tuned binaries don't always outperform the POWER5-balanced binaries on the POWER6 platform (as the data in Figure 4-1 show). The best performance for this case (and several other cases) seems to come from the options of set 2. But this is not always the case. NPB, for example, is much faster using POWER6 tuning. We recommend trying both balanced and POWER6 tuning for now.

5 MPI Performance Optimization on POWER6

In this Chapter, we will present the use of an MPI environment in POWER6 compute clusters. In many cases, there are MPI and other system level settings that can be used to improve application-level performance. We provide an overview of node and cluster level architecture before describing the use of different options available to a user to optimize application performance. Performance measurements on different HPC benchmarks are provided as well as the settings that were used for obtaining those results.

The building block for high-end POWER6 compute clusters is an IBM Power 575 system. The 575 system (or node) is 2U in height and has 32 POWER6 processor cores running at 4.7 GHz. Large compute clusters can be built by using an Infiniband switch network and multiple 575 nodes. In Figure 5.1, a high level view of the 575 system architecture is shown. There are 16 POWER6 processor modules (DCM) in a system. Each DCM (Dual Chip Module) has a POWER6 processor chip (p6) and an L3 cache chip. The processor chips are connected together by different types of fabric busses. As shown in the figure, a "Quad" constitutes 4 processor chips connected by "X", "Y" and "Z" busses (shown in black). Each X, Y or Z bus is 2B wide, runs at half the speed of the processor core and carries both data and address traffic. Processor chips in each Quad are connected to processor chips in other Quads using "A" and "B" fabric busses which are 4B wide (shown in blue and red). Each chip has one memory controller enabled and is connected to 4 DIMM slots that can be populated with DIMMs of different density. The 575 can support 1GB, 2GB and 4GB DIMMs. Each node has to be populated by the same type of DIMMs. A maximally configured 575 node can have up to 256GB of RAM. Each 575 node can have 2 Infiniband Galaxy-2 adapter cards for clustering. To get a better understanding of MPI performance, we will look first on a single node 575 and then on a multi-node 575 cluster. In general, there are several options presented below that can show significant performance improvements.

Figure 5-1 IBM Power 575 System Architecture



5.1 POWER 6 Cluster Interconnect Overview

IBM has chosen the QLogic™ 9000 family of multi-protocol fabric directors as the basis for the Infiniband interconnect of its Power 575 clusters. The 9000 Fabric Directors are designed to maximize cluster and grid computing interconnect performance while simplifying and reducing the cost of operating a data center.

The family of QLogic switches scale to support as few as 10 nodes using a single 2U chassis to multi-thousand node networks using multiple 288 port Fabric Director systems working in concert as a single high performance virtual fabric. All chassis types are slot-independent and each supports flexible amount of switching capacity and port density – as many as 288 ports and 11.52 Tb/s of switching throughput in a single chassis.

Each Power 575 compute node supports up to two 4 port IBM Infiniband Host Channel Adapters (HCA). Each HCA has dual 4x DDR ports to connect into the QLogic fabric. The HCA is IBM designed based on IBM's high performance Galaxy2 Infiniband chip and is local bus attached through the GX++ bus directly to POWER6 (see Figure 5-2). Using two adapters, a 575 system can have up to 8 ports providing 15 GB/s unidirectional sustained total bandwidth. In many configurations, using 4 ports per compute node is sufficient. In the 4 port configuration, it is still recommended to use two adapters and two ports each versus one adapter per node with four ports.

Figure 5-2 IBM Infiniband HCA with 4 IB DDR links

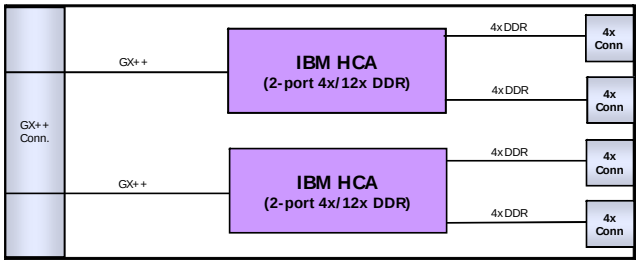
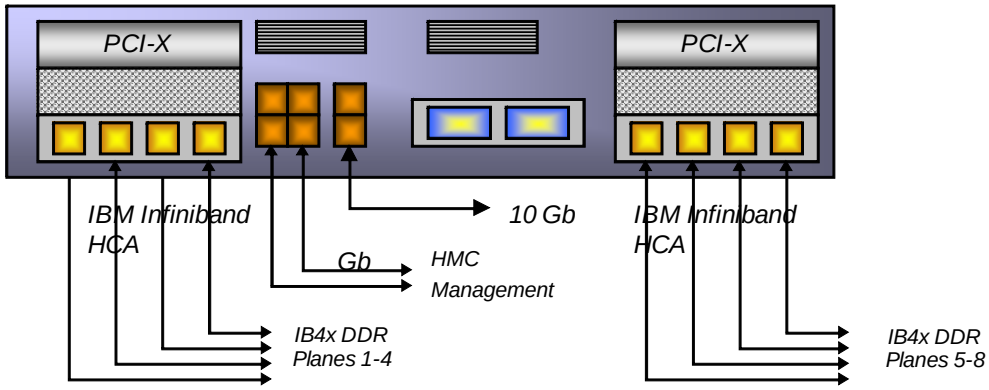


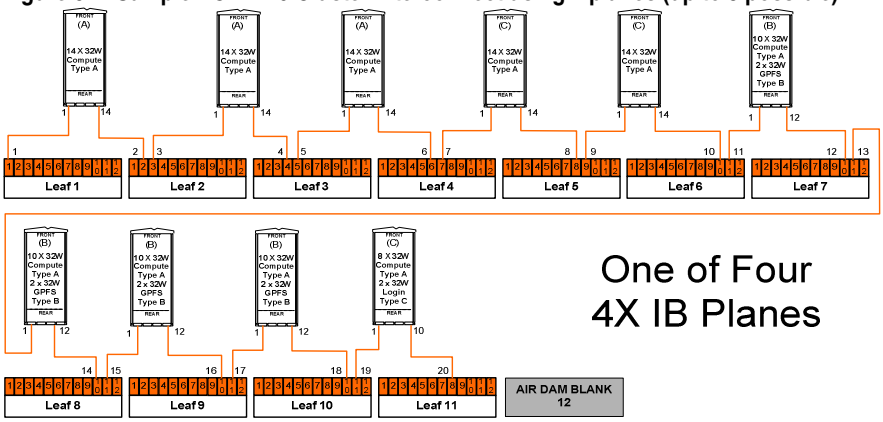
Figure 5-3 Rear view of a Power 575 system with 8 4x DDR Infiniband connections



Interconnect between nodes and switches is organized in planes where each port of a single node usually connects to a different switch or plane. For example, the first port of every node would connect to the same switch. See Figure 5-4.

The IBM Infiniband HCA adapter has a latency of 3.4 microseconds for MPI communication messages. Each link can sustain a unidirectional bandwidth of 1.9 GB/s, and a bidirectional bandwidth of 3.8 GB/s. These bandwidths are achievable by a single MPI task if memory is allocated optimally.

Figure 5-4 Sample POWER6 Cluster Interconnect using 4 planes (up to 8 possible)



5.2 MPI Performance Considerations on 575

Optimal performance on the 575 may be achieved by keeping MPI tasks close to their memory. This section will focus on how to use the latest LoadLeveler and IBM PE (Parallel Environment) affinity features to constrain MPI tasks to run on cores or MCMs (the choice may depend on the application, and what mode the 575 is booted to). Note that the term MCM does not always refer to a multichip module. In LoadLeveler and PE, the use of MCM refers to an affinity domain that equates to a single 'quad' (8 cores) of a 575 node. MCM can also refer to an affinity domain in other contexts. For example, setting the AIX environment variable MEMORY_AFFINITY=MCM requests AIX to allocate memory within a local affinity domain, but the size of the domain can depend on how the 575 is booted (for the 575, in chip affinity mode, the domain is a single DCM chip, but in node affinity mode, the domain spans a quad, or 8 cores).

On POWER5 machines, the use of binding and/or rsets has often been required to achieve the best HPC performance, and it's likely that the 575 will see even greater benefits in this area.

Note the term rset is AIX-specific but we'll use it here to refer to both AIX rsets and Linux cpusets. The Linux LoadLeveler job scheduler affinity implementation uses Linux cpusets, though LoadLeveler still uses the same 'rset' keywords for both Linux and AIX.

The use of affinity rsets is preferred over bindprocessor calls since rsets are more flexible and there can be difficulties in mapping between bindprocessor IDs, logical CPU IDs, physical CPUs, and affinity domains. The latest levels of the LAPI communication protocol support achieving affinity when bindprocessor is used but rset binding is generally preferred.

5.2.1 LoadLeveler Affinity Settings

In an IBM scheduler solution, LoadLeveler should provide the required rset support.

- In order to use the LoadLeveler affinity scheduling support, the first step is to set the **rset_support** keyword to value **rset_mcm_affinity** in the LoadLeveler configuration file.

- After starting LoadLeveler with affinity scheduling support, the usage and availability of MCMs in the entire cluster can be queried using **llstatus –M** command.

The most important JCF keyword is the **rset** keyword which should be set to **rset_mcm_affinity** to force an rset to be created for each MPI task.

#@rset = rset_mcm_affinity

Each task will be constrained to run in an MCM affinity domain if no other options are set, but the **mcm_affinity_options** keyword will probably also need to be set to get good performance. With the default value, tasks will accumulate on the same MCM, leaving other MCMs idle. This may be good for a small job that doesn't need more resources than a single MCM provides, but generally round-robin scheduling of the tasks across MCMs is preferable. A suggested value for **mcm_affinity_options** for MPI jobs running on the 575 is:

#@mcm_affinity_options=mcm_distribute mcm_mem_req mcm_sni_none

The mcm_affinity_options keyword should define **one** of each of these three types of affinity options: **task_mcm_allocation**, **memory_affinity**, and **adapter_affinity**, which can have the following possible values:

task_mcm_allocation options:

(Generally should be set to mcm_distribute unless all tasks running on an 575 node can fit constrained within a single MCM)

gmcm_accumulate (tasks are placed on the same MCM when possible)
mcm_distribute (tasks are round-robin distributed across available MCMs)

memory_affinity options:

(the default value should suffice here, but it's suggested that MEMORY_AFFINITY=MCM be exported into the job's environment to be sure that local memory requests are made)

mcm_mem_none (the job has no memory affinity requirement.)
mcm_mem_pref (memory affinity preferred)
mcm_mem_req (memory affinity required)

adapter_affinity options:

mcm_sni_none (no adapter affinity requirement; recommended for the 575)
mcm_sni_pref (adapter affinity preferred)
mcm_sni_req (adapter affinity required)

It is recommended that all IB links be defined to LoadLeveler, and that the network specification "sn_all" be made for User Space jobs. This will request LoadLeveler to assign windows on each link for use by the parallel job.

On POWER6 Infiniband (IB) systems, the adapter affinity options should not be needed. In the POWER4/5 HPS Interconnect design, the adapter affinity option was intended to allow a task to be constrained to run within the same affinity domain as the HPS link that it was assigned by LoadLeveler. In the POWER6 IB design, however, the LAPI (Low Level API) protocol will determine which adapter is the best choice to send messages over when multiple adapters are available for the job.

For single link (non-striped) smaller FIFO messages, the LAPI protocol will attempt to send over the adapter in the same quad if possible. Each task will typically send FIFO

messages over one link, but may receive over any available link.

For maximum bandwidth, RDMA traffic will be striped equally across all available links. A maximum of eight links are supported on the 575, which will use the available GX++ buses evenly for IB traffic. For cases in which four links are used, the four links should be plugged using two separate IBM Infiniband HCA adapters, so that each link has one GX++ bus connection to each 'quad' (8 cores), again allowing an even distribution of load on the GX++ buses.

With LoadLeveler 3.5 additional rset affinity domains beyond **mcm** have been added via a new JCF **task_affinity** keyword, which allows constraining a task to run on a number (**x)** of cores or CPUs, e.g.

#@task_affinity=core(x)
Or
#@task_affinity= cpu(x)

The core keyword specifies that each task in the job is bound to run on as many processor cores as specified by **x.** The CPU keyword indicates that each task of the job is constrained to run on as many logical CPUs as defined by **x**. The CPUs/cores will not be shared between tasks scheduled by LoadLeveler if scheduling affinity is enabled.

An optional new JCF keyword, **cpus_per_core**, allows for customization of the number of threads per core used by an MPI task:

#@cpus_per_core=x

Note that, for a POWER6 with SMT enabled, only values of 1 and 2 make sense for **x**. This defines how many SMT threads the task can run on for every core it is assigned.

When the new LoadLeveler **task_affinity** and **cpus_per_core** options are used, the **rset** and **mcm_affinity_options** can be used in combination to request mcm specific affinity options in addition to the processor core affinity options.

Example

- The following example is for one thread per core in ST mode and two threads per core in SMT mode:

  #@task_affinity=core(1) for ST mode
  #@task_affinity=cpu(1) for SMT mode.

5.2.2 Parallel Environment Affinity Support

IBM Parallel Environment 5.1 (PE) has extended its affinity support with additional rsets for cases that correspond to the latest LoadLeveler affinity support. PE's support of affinity domains is controlled by the **MP_TASK_AFFINITY** environment variable. It provides roughly equivalent functionality to LoadLeveler's JCF keywords, except PE can provide only one core or one logical CPU to each MPI task.

The **MP_TASK_AFFINITY** settings are ignored for batch jobs. To give a batch job affinity, the appropriate LoadLeveler JCF keywords should be used.

For interactive jobs, POE will query the LoadLeveler API to determine if the resource manager provides affinity support at the requested level. When running a version of LoadLeveler with full affinity support, PE will simply convert the requested **MP_TASK_AFFINITY** environment variable to the appropriate JCF settings as follows:

Table 5-1 POE Environment Variable Settings and corresponding LL JCF lines generated

POE Environment Variable Setting:	LoadLeveler JCF Lines that will Be Generated:
MP_TASK_AFFINITY = MCM	**#@rset = rset_mcm_affinity**
MP_TASK_AFFINITY = CORE	**#@rset = rset_mcm_affinity** **#@task_affinity = core(1)**
MP_TASK_AFFINITY = CPU	**#@rset = rset_mcm_affinity** **#@task_affinity = cpu(1)**
MP_TASK_AFFINITY = SNI	**#@rset = rset_mcm_affinity** **#@mcm_affinity_options = MCM_SNI_PREF MCM_DISTRIBUTE**
MP_TASK_AFFINITY = list	**Ignored**

The **MP_TASK_AFFINITY=list** approach defines a list of MCM rset numbers that have already been created.

The **MP_TASK_AFFINITY=SNI** setting is probably not needed for the 575.

5.2.3 Binding Tools for Affinity

In addition to the standard affinity options mentioned above, there are also tools available to help user easily take advantage of binding based on type of workload being run, pure MPI, MPI + OpenMP or pure OpenMP. For more details, see the tools affinity web page[11].

5.2.4 Additional MPI Environment Settings

For better performance, the following environment variables can be used on top of the LoadLeveler or PE affinity settings:

```
export MP_SYNC_QP=YES
export MP_RFIFO_SIZE=16777216
export MP_SHM_ATTACH_THRESH=500000
export MP_EUIDEVELOP=min
export MP_USE_BULK_XFER=yes
```

RDMA specific tunables:

```
export MP_RDMA_MTU=4K
export MP_BULK_MIN_MSG_SIZE=64k
export MP_RC_MAX_QP=8192
```

MP_SYNC_QP=YES

On IB systems, QP information must be exchanged between two tasks before messages can be sent. For FIFO/UD traffic, it's recommended that the exchange of this information be done in MPI_Init() by setting MP_SYNC_QP=YES. By forcing this QP exchange to occur up-front, some variation in communication performance can be eliminated.

MP_RFIFO_SIZE=16777216

The default size of the receive FIFO used by each MPI task is 4MB. Larger jobs are recommended to use the maximum size receive FIFO by setting

[11] http://loadl.farfalladesign.com/Binding%20in%20AIX.htm

MP_RFIFO_SIZE=16777216.

MP_SHM_ATTACH_THRESH=500000

LAPI has two modes of sending shared memory messages. For smaller messages slot mode is used to copy messages from one MPI task to another. For larger messages, it's possible to map the shared memory segment of one task to another, thereby saving a copy at the cost of some overhead of attaching. The MP_SHM_ATTACH_THRESH variable defines the minimum size message for which attach mode is used. Depending on the type of job different cross-over points may provide optimal performance, but 500000 is often a reasonable starting point when tuning this value. The default depends on how many tasks are running on the node.

MP_EUIDEVELOP=min

The MPI layer will perform checking on the correctness of parameters according to the value of MP_EUIDEVLOP. As these checks can have a significant impact on latency, when not developing applications it is recommended that MP_EUIDEVELOP=min be set to minimize the checking done at the message passing interface layer.

MP_USE_BULK_XFER=yes

Setting MP_USE_BULK_XFER=yes will enable RDMA. On IB systems using RDMA will generally give better performance at lower task counts when forcing RDMA QP information to be exchanged in MPI_Init() (via setting LAPI_DEBUG_RC_INIT_SETUP=yes). When the RDMA QP information is not exchanged in MPI_Init(), there can be delays due to QP information exchange until all tasks have synced-up.

The benefit of RDMA depends on the application and its use of buffers. For example, applications that tend to re-use the same address space for sending and receiving data will do best, as they avoid the overhead of repeatedly registering new areas of memory for RDMA.

RDMA mode will use more memory than pure FIFO mode. Note that this can be curtailed by setting MP_RC_MAX_QP to limit the number of RDMA QPs that are created.

MP_BULK_MIN_MSG_SIZE=64k

The minimum size message used for RDMA is defined to be the maximum of MP_BULK_MIN_MSG_SIZE and MP_EAGER_LIMIT. So if MP_EAGER_LIMIT is defined to be higher than MP_BULK_MIN_MSG_SIZE, the smallest RDMA message will be limited by the eager limit.

MP_RC_MAX_QP=8192

This variable defines the maximum number of RDMA QPs that will be opened by a given MPI task. Depending on the size of the job and the number of tasks per node, it may be desirable to limit the number of QPs used for RDMA. By default, when the limit of RDMA QPs is reached, future connections will all use FIFO/UD mode for message passing.

5.3 Improved Network Interface Fault Detection and Failover

Prior to the POWER6 release of MPI/LAPI, detection of adapter problems was done through LAPI subscribing to event notification through the RSCT (Reliable Scalable Clusters Technology) component. RSCT detects network interface failures by doing IP (UDP) packet heart-beating in a ring topology. Starting with the release of P6 IB systems, LAPI detects adapter failures by 'on-demand pings' to interfaces that LAPI needs to retransmit to. The term 'ping' here does not refer to the traditional ICMP ping,

but instead is a protocol message (e.g. user-space packets are sent when the running job is using user-space communication).

The failover done by the LAPI protocol only works when LAPI is using more than one link to stripe data over. When there is more than one link available, LAPI can stop using links that are found to cause problems. LAPI can ensure that the running job will stop using the problem links, 'failing-over' to only send data over the working link(s).

The new design has several benefits. It is on-demand, removes external dependencies on RSCT, and requires no additional cycles in the protocol layer unless a task starts retransmitting. When connectivity problems occur, only tasks directly involved are impacted. Also the 'ping' packets more directly check the actual communication paths used by the MPI job, and the new code works for any interconnect and striping level.

5.4 Performance Results

High Performance Computing has innumerable applications and equally numerous benchmarks. It is always the best to evaluate performance of an HPC system with one's own application and input data set. However, it is not feasible in practice due to logistical impediments benchmarking using one's own workload. We provide in the ensuing sections, performance results on several benchmark suites that should only be viewed as providing guidance as to the capability of a 575 compute cluster. As we saw in the previous sections, there are many approaches to optimizing application performance on a 575 compute cluster and the recommended step in the evaluation process is to use a real application and a representative data set for benchmarking.

5.4.1 HPCC Benchmarks

High Performance Computing Challenge (HPCC) benchmark comprises several tests. We provide results on 6 of the tests that exercise different types of computation and communication a cluster of 575 nodes connected by an Infiniband switch network described in Section 5.1.

HPCC 1.2.0 was used on the 575 cluster in which each 575 node had 64GB of RAM and two Infiniband HCAs with 4 ports per HCA connected to the Infiniband switch. Compiler flags and the environment settings used for the benchmark are shown in Appendix C.

In the table below, RR lat and RR bw are the Random Ring latency and bandwidth tests respectively.

Table 5-2 HPCC Benchmark Performance on Power 575 Cluster

Cluster Size # of 575 nodes	**HPL** GFLOPs	**PTRANS** GB/s	**RandomAccess** GUPs	**MPIFFT** GFLOP/s	**RR lat** usecs	**RR bw** GB/s
1	423	9.3	0.12	20.7	4.16	2.91
2	840	14.1	0.23	32.6	4.93	0.49
4	1682	19.0	0.42	59.3	5.12	0.29
8	3292	37.0	0.76	105.0	5.47	0.23
16	6570	73.4	1.34	179.5	8.14	0.21
32	12780	122.4	2.43	336.9	8.57	0.20
64	25740	259.3	4.54	630.6	8.91	0.18

5.4.2 NAS Parallel Benchmarks

NAS Parallel Benchmarks is another suite that is commonly used in HPC. We used the same 575 cluster described in Section 5.5.2 and Class D size benchmarks. **cg** is a conjugate gradient method based benchmark which results in sparse matrix-vector

computations stressing memory bandwidth in a 575 node. **ft** on the other hand uses a 3D FFT method exercising all-to-all communication pattern in the cluster. **mg** is the multi-grid benchmark while **lu** and **sp** are pseudo-application benchmarks representative of structured grid computations used in CFD. Performance (Mop/s) as reported by individual, parallel benchmark is shown in the table below.

Table 5-3 NAS Parallel Benchmark Performance on Power 575 Cluster

Cluster Size # of 575 nodes	**MPI** Tasks	**cg.D** Mop/s	**ft.D** Mop/s	**lu.D** Mop/s	**mg.D** Mop/s	**sp.D** Mop/s
1	32 (a)	8714	(*)	69946	54402	20606
2	64	16049	(*)	214061	117028	59832
4	128 (b)	29174	88832	265205	218623	113175
8	256	45299	166415	504727	416813	234856

NOTES:

(a) – 25 tasks for SP
(b) – 121 tasks for SP
(*) – did not run due to insufficient memory in each 575 node

5.4.3 SPEC MPI2007 Benchmarks

MPI2007 is a suite of MPI application benchmarks introduced in 2007 by the SPEC organization for evaluating floating-point, compute intensive performance on clusters and SMP systems. The benchmark suite comprises 13 different benchmarks representing various application areas in HPC including CFD, quantum chemistry, molecular dynamics, etc. Input data sets in the first release of the benchmark are considered "medium" that provide limited scalability on many of the benchmarks.

Single and dual-node 575 results are published and the results are available on the SPEC website[12].

Power 575 can provide significant performance benefit on many different types of applications when SMT mode is turned on. In the table below, we compare the performance benefit due to SMT on each of the individual application benchmarks from the MPI2007 suite. All the results are for a single node of a 575. ST in the table represents performance (normalized with respect to a reference system) of an application with the 575 in a single thread mode. In ST mode, each MPI application was run using 32 tasks per 575 node or one MPI task per POWER6 core. In SMT mode, each application was run using 64 tasks per 575 node or 2 MPI tasks per POWER6 core.

On 12 out of 13 applications, SMT provided performance benefit ranging from 15% to 71%. Only 130.socorro showed a small degradation (10%) in performance due to SMT. In general, applications that profile on the two ends of the performance characteristic spectrum, either extremely core bound or extremely memory bandwidth bound will see the least benefit from SMT.

Table 5-4 Performance Benefit with SMT on MPI Applications

MPI2007 Benchmark	**Performance (rate) in different system modes**		**Performance benefit with SMT** (%)
	ST	**SMT**	
104.milc	2.74	4.21	54
107.leslie3d	6.91	7.96	15
113.GemsFDTD	8.92	10.3	15
115.fds4	4.75	6.52	37
121.pop2	4.86	5.81	20

[12] http://www.spec.org/mpi2007/results/mpi2007.html

122.tachyon	1.82	3.12	71
126.lammps	3.86	6.46	67
127.wrf2	5.92	8.88	50
128.GAPgeofem	6.34	9.03	42
129.tera_tf	2.26	3.63	61
130.socorro	14.4	12.9	-10
132.zeusmp2	3.29	5.13	56
137.lu	6.65	8.30	25

6 High Performance Libraries

6.1 ESSL

6.1.1 Matrix Multiply

Table 6-1 shows performance results (in GFlops/second) for the standard ESSL matrix multiply kernels. The version used is ESSL 4.4. The runs were done on a 32-core 4.7 GHz Power 575 booted in ST mode. Large pages (16 MB) were used. A 4000x4000 sized matrix was used.

Table 6-1 ESSL 4.3 - Performance on POWER6

	SGEMM	DGEMM	ZGEMM	CGEMM
1 thread	15.7	15.4	16.7	17.0
4 threads	62.3	61.3	66.4	67.5

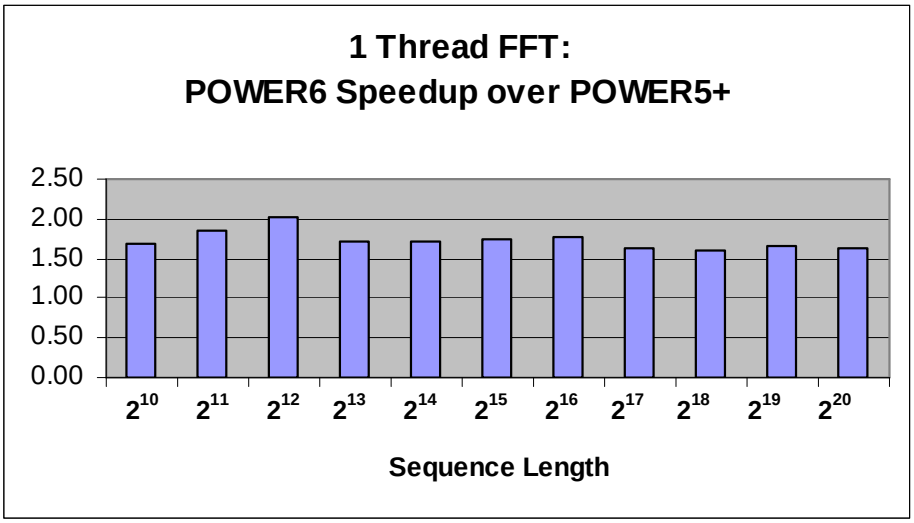
6.1.2 SCFT (single precision complex FFT)

These results were obtained using ESSL 4.3.

Figure 6-1Figure 6-1 examines the performance of SCFT on the POWER5+ and POWER6 for various sequence lengths ranging from 2^{10} to 2^{20}.

The POWER5+ system used was a 1.9 GHz p575+ and the POWER6 system was a 4.7 GHz 4W p570.

Figure 6-1 POWER6 Speedup of ESSL routine SCFT running 1 thread for a range of sequence lengths



The results show POWER6 performance generally exceeds 1.5 times that of the POWER5+.

6.2 MASS/MASSV

Mathematical Acceleration Subsystem (MASS) for AIX® consists of libraries of tuned

mathematical intrinsic functions. MASSV is a library of vectorized versions of the intrinsic functions.

These libraries:

- Include both scalar and vector functions.
- Are thread-safe.
- Support both 32-bit and 64-bit compilations.
- Offer improved performance over their corresponding standard system library (**libm**) routines.
- Are intended for use in C, C++, and FORTRAN applications where slight differences in accuracy or handling of exceptional values can be tolerated.

On POWER6, the easiest approach to speeding up the intrinsic math functions also gives the best performance; use the compiler flags **-O3 –qhot**. There is no need to hand-code the vector math intrinsics. The best MASS lib to use is the one optimized for POWER6.

6.2.1 Details

The following charts look at the different ways that mathematical functions can be invoked from a program:

- Use the libm math routines provided with AIX
- Use the MASS library routines
- Use the vector MASS library routines (presented for various vector lengths)
- Use the compiler to generate customized versions of the vector intrinsics.

Figure 6-2 MASS Library Performance and compiler-generated routines on POWER5+ and POWER6

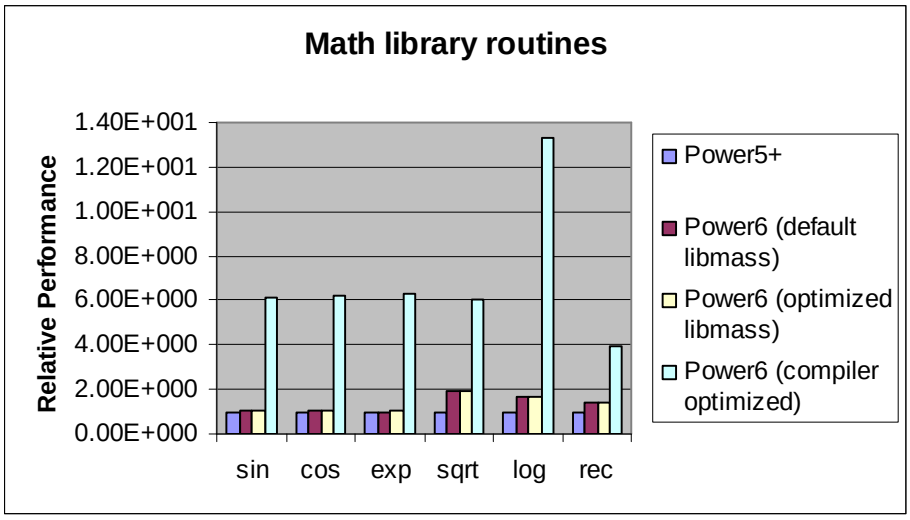


Figure 6-3 Relative performance of math intrinsic function options

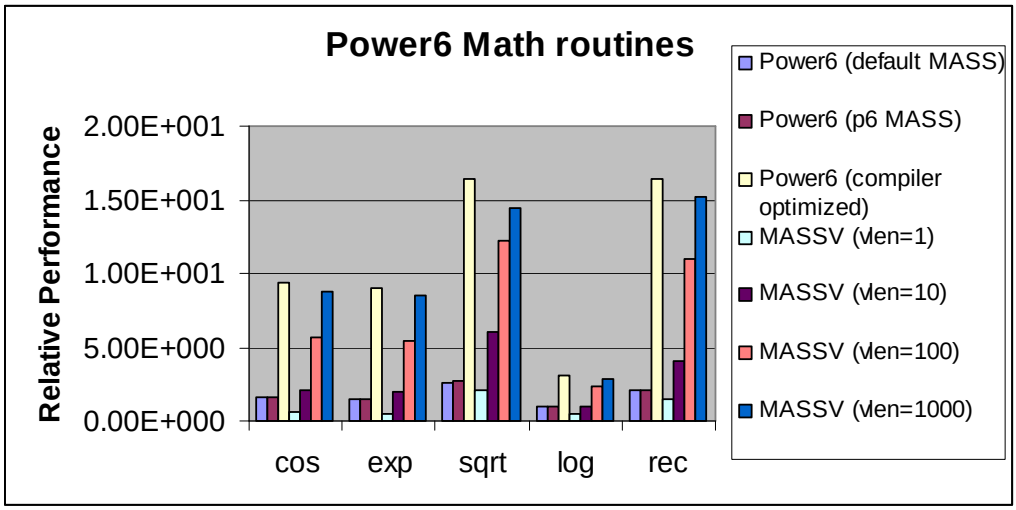


Figure 6-3 above shows normalized performance ratios (relative to the POWER6 libm library) for various sources for math function options. All of the MASSV data is from the POWER6-optimized vector MASS library.

6.2.2 Summary

On POWER6, MASS library can improve the performance of intrinsic mathematical functions up to 2.67 times the system versions supplied in **libm.a**. Writing code to use the MASSV library can further improve performance by a factor of 10. But the easiest approach also gives the best performance; use compiler flags **-O3 –qhot.** These compile options instruct the compiler to automatically replace intrinsic functions with vector equivalents, choosing the optimal vector length for the problem. Any intrinsics that cannot be vectorized are then replaced at link time with the better performing scalar versions from the MASS library. In most cases, the compiler creates faster code than what is

achieved by using MASSV, and without the need to rewrite any code.

See here[13] for a reference on MASS.

[13] http://www-306.ibm.com/software/awdtools/mass/aix/mass-aix.html

7 POWER6 instruction set differences

7.1 New instructions

The following is a subset of new instructions supported in POWER6 that were not supported in POWER5:

- **frxe, frxes, frsqrtxe, frsqrtxes**: new higher precision (15-bit) estimate instructions are added to the architecture to improve the performance of software emulations of divide and square root; these instructions use the new lookup tables and linear approximation circuits
- 64 bit FPSCR instructions.
- VMX/AltiVec instruction set
- **cmpb** - Byte compare. Supports Database, parsing, and string comparison (bioinformatics) tasks.
- **prtyd** - Byte Parity.
- Variance of dcbfl that only flushes L1 cache. Enable hardware support of block moves.

Appendix A: POWER5+ and POWER6 Hardware Comparisons

A.1 Core Features[14]

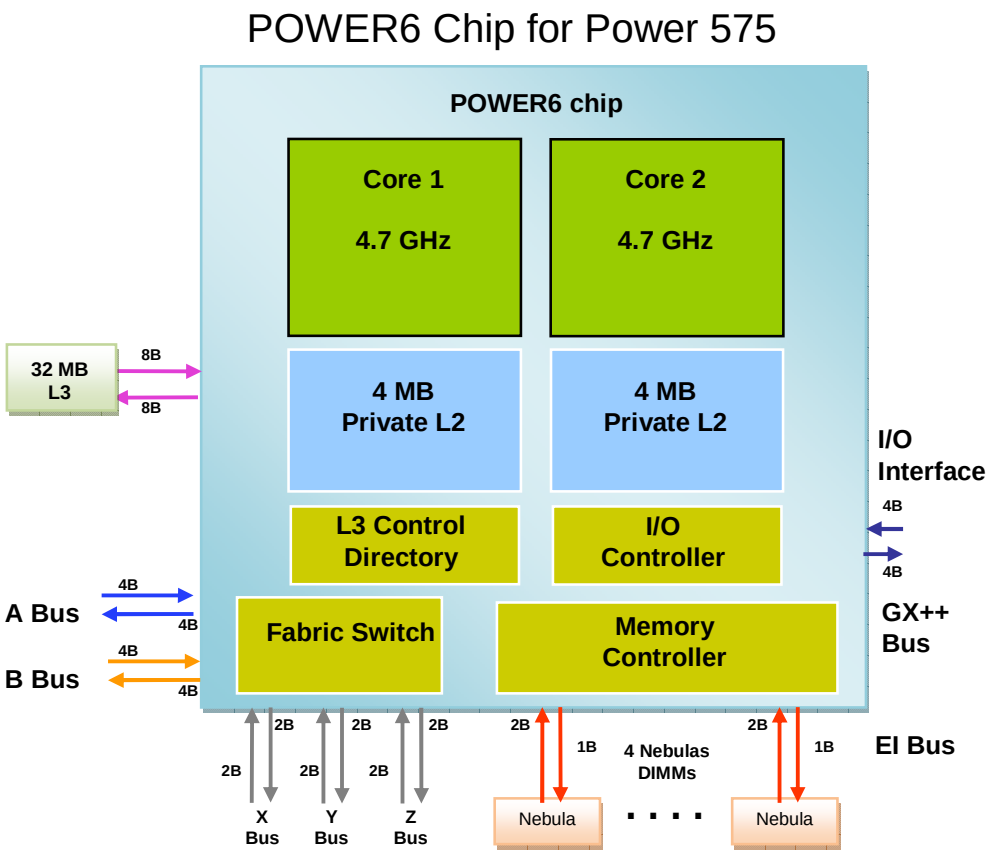
Feature	POWER5+	POWER6
Registers	120 FPR/GPR	64 FPR/GPR (no renaming)
Cache	64K 2W I Cache 32K 4W D Cache (2RD + 1WR) Dedicated L2 reload for I and D 32B reload bus at core frequency 1.88 MB L2 shared by 2 cores per chip 36 MB L3	64K 4W I Cache 64K 8W D Cache (2RD/1WR) Shared L2 reload bus for I and D 32B reload bus at core frequency 4 MB L2 not shared 32 MB L3
Functional Units	2FX, 2LS, 2FP, 1BR, 1CR	2FX, 2LS, 2FP, 1BR/CR, 1DP, 1VMX/AltiVec
FPU Pipe line	2-nine stage (6 execution)	2-eight stage (6 execution)
Threading	2 thread SMT Alternate instruction fetch Alternate dispatch (5 instructions)	2 thread SMT Priority-based dispatch Simultaneous dispatch from 2 threads (7 instructions)
Instruction Dispatch	5 instruction dispatch, branch allowed at end of group 1 thread per cycle In flight: ST – 191 SMT – 175(1 thread) 215(2 threads)	5 Instruction dispatch per thread 7 Instruction dispatch for 2 threads (SMT) 1 branch at any location in group 2 threads per cycle In flight: 1 thread - 120 2 threads – 184
Instruction Issue	2FX, 2LS, 2FP, 1BR, 1CR does not reject instructions, but does recycle instructions for re-execution	2FX, 2LS, 2FP/1DP, 1BR/CR VMX/AltiVec uses FPQ and VIQ
Rename	Yes	No Load Target Buffer (up to 20 loads)
Translation	I-ERAT = 128 entries, 2W (4K,64K page) D-ERAT = 128 entries, fully associative (4K,64K page) SLB = 64 entries per thread TLB = 2K, 4W 65 bit VA, 50 bit RA Page size = 4K, 64K, 16M, 16G	I-ERAT = 128 entries, 2W (4K,64K page) D-ERAT = 128 entries, Fully set associative (4K,64K, 16M page) SLB = 64 entries per thread 68 bit VA, 48 bit RA Page size = 4K, 64K, 16M, 16G

[14] See IBM Journal of R&D, Volume 51, November 6, 2007. for more detail (http://www.research.ibm.com/journal/rd51-6.html).

Appendix B: IBM System Power 575 Compute Node

The POWER6 chip has many memory and bus options across system implementations. Figure B.1 shows POWER6 as implemented in the IBM Power 575 system.

B.1 Specific application of POWER6 chip in the 575 compute node



Appendix C: Compiler Flags and Environment Settings for HPCC Benchmarks

Following are the compiler flags and the environment settings used in the HPCC benchmark runs on the 575 cluster system (See Section 5.4.1).

CC = mpcc_r -q64

CCFLAGS = $(HPL_DEFS) -O3 -qarch=pwr6 -qtune=pwr6 -I$(TOPdir)/hpl/include -DIBM_RUN_INDIVIDUAL_KERNELS -DFFTE_NBLK=8 -DRA_SANDIA_OPT2 -DHPCC_FFT_235

LINKER = mpcc_r -q64 -bdatapsize:64k -bstackpsize:64k

Calls to the AIX “disclaim” subroutine were added at the ends of the PTRANS and HPL benchmarks to try to prevent HPCC from failing due to an apparent lack of memory. In general, physical memory is not immediately released when a program “free”s it.

The following MPI environment variables were set (RDMA was not used because insufficient RC queue pairs are available with 64 nodes, 2048 tasks)

```
export LAPI_DEBUG_STRIPE_ENABLE_PING=no
export MP_FIFO_MTU=4096
export MP_CLOCK_SOURCE=AIX
export MP_EAGER_LIMIT=65536
export MP_CSS_INTERRUPT=no
export MP_SHM_ATTACH_THRESH=400000
export MEMORY_AFFINITY=MCM
export MP_EUIDEVELOP=min
export MP_INFOLEVEL=2
export MP_PROCS="$1"
export MP_EUILIB=us
export MP_EUIDEVICE=sn_all
export LAPI_DEBUG_ENABLE_AFFINITY=yes
export MP_SHARED_MEMORY=yes
export MP_WAIT_MODE=poll
export MP_SINGLE_THREAD=yes
export MP_TASK_AFFINITY=CORE
```

Appendix D: Acknowledgements

We would like to thank the following people who provided many of the HPC benchmark results shown in the paper.

- Farid Parpia
- Frank Johnston
- Giri Prabhakar
- Joan McComb

Appendix E: Notices

IBM Corporation
Marketing Communications, Systems Group
Route 100, Somers, New York 10589
Produced in the United States of America
March 2009, All Rights Reserved

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM

products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

More details can be found at the IBM Power Systems home page[15].

[15] http://www.ibm.com/systems/p

Appendix F: Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

1350™
AIX 5L™
AIX®
alphaWorks®
Ascendant®
BetaWorks™
BladeCenter®
CICS®
Cool Blue™
DB2®
developerWorks®
Domino®
Enterprise Storage Server®
Enterprise Workload Manager™
eServer™
Express Portfolio™
FlashCopy®
GDPS®
General Parallel File System™
Geographically Dispersed Parallel Sysplex™
Global Innovation Outlook™
GPFS™
HACMP™
HiperSockets™
HyperSwap™
i5/OS®
IBM Process Reference Model for IT™
IBM®
IntelliStation®
Lotus Notes®
Lotus®
MQSeries®
MVS™
Netfinity®
Notes®
OS/390®
Parallel Sysplex®
PartnerWorld®
POWER™
POWER4™
POWER5™
POWER6™
PowerExecutive™
PowerPC®
PowerVM™
PR/SM™
pSeries®
QuickPlace®
RACF®
Rational Summit®
Rational Unified Process®
Rational®
Redbooks®
Redbooks (logo) ®
RS/6000®
RUP®
S/390®
Sametime®
Summit Ascendant™
Summit®
System i™
System p™
System Storage™
System x™
System z™
System z10™
System/360™
System/370™
Tivoli®
TotalStorage®
VM/ESA®
VSE/ESA™
WebSphere®
Workplace™
Workplace Messaging®
X-Architecture®
xSeries®
z/OS®
z/VM®
z10™
zSeries®

The following terms are trademarks of other companies:

- AltiVec™ is a trademark of Freescale Semiconductor, Inc.
- AMD™, AMD Opteron™, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.
- InfiniBand™, and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.