Overview of BlueGene/L System Software

Manish Gupta  José Moreira
IBM T. J. Watson Research Center
BlueGene/L System Software Team


- B de Supinski, D Dossa, L Kissel, M Seager, R Yates (LLNL)
Outline

- Design goals
- Summary of system software design
- Early performance results

Later talks:
- MPI design and implementation
- Programming for multi-node performance
- Compiler design, programming for single-node performance
Software Design Goals

- Familiar software development environment and programming models

- Scalability to $O(100,000)$ processors
  - Performance
  - Reliability, Availability, Serviceability
Complete BlueGene/L System at LLNL
Blue Gene/L System Software Organization

- **Console**
- **DB2**
- **Scheduler**
- **Service Node**
- **MMCS**
- **Front-end Nodes**
- **Control Ethernet**
- **Functional Ethernet**
- **File Servers**
- **IDo chip**
- **I2C**
- **I/O Node 0**
  - **Linux**
  - **cioid**
  - **CNK**
- **C-Node 0**
  - **CNK**
- **C-Node 63**
  - **CNK**
- **I/O Node 1023**
  - **Linux**
  - **cioid**
  - **CNK**
- **Pset 0**
- **Pset 1023**

**Networks:**
- **tree**
- **torus**

**Host Systems:**
- **Front-end Nodes**
- **File Servers**

**Control Systems:**
- **Service Node**
- **MMCS**
- **IDo chip**
- **I2C**

**Software:**
- **Linux**
- **cioid**
- **CNK**
Software Design Summary

- **Familiar software development environment and programming models**
  - Fortran, C, C++ with MPI
    - Full language support
    - Automatic SIMD FPU exploitation
  - Linux development environment
    - User interacts with system through FE nodes running Linux – compilation, job submission, debugging
    - Compute Node Kernel provides look and feel of a Linux environment – POSIX system calls (with restrictions)
  - Tools – support for debuggers, hardware performance monitors, trace based visualization

- **Scalability to $O(100,000)$ processors**
Software Design Summary

- Familiar software development environment and programming models
- **Scalability to \( O(100,000) \) processors – through Simplicity**
  - **Performance**
    - Strictly space sharing - one job (user) per electrical partition of machine, one process per compute node
      - Dedicated processor for each application level thread
        - Guaranteed, deterministic execution
        - Physical memory directly mapped to application address space – no TLB misses, page faults
    - Efficient, user mode access to communication networks
      - No protection necessary because of strict space sharing
    - Multi-tier hierarchical organization – system services (I/O, process control) offloaded to IO nodes, control and monitoring offloaded to service node
      - No daemons interfering with application execution
      - System manageable as a cluster of IO nodes
  - **Reliability, Availability, Serviceability**
    - Reduce software errors - simplicity of software, extensive run time checking option
    - Ability to detect, isolate, possibly predict failures
Programming Models for Compute Node

- **Communication coprocessor mode**: CPU 0 executes user application while CPU 1 handles communications
  - Preferred mode of operation for communication-intensive and memory bandwidth intensive codes
  - Requires coordination between CPUs, which is handled in libraries

- **Computation offload mode**: CPU 1 executes some parts of user application offloaded by CPU 0, in addition to communication
  - Can be selectively used for compute-bound parallel regions
  - Asynchronous coroutine model (co_start / co_join)
  - Need careful sequence of cache line flush, invalidate, and copy operations to deal with lack of L1 cache coherence in hardware

- **Virtual node mode**: CPU0 and CPU1 handle both computation and communication (future work)
  - Two MPI processes on each node, one bound to each processor
  - Distributed memory semantics – lack of L1 coherence not a problem
Math Library

■ Subset of ESSL (or LAPACK) being developed
  ❖ Mainly dense matrix kernels – DGEMM, DGEMV, DDOT, DAXPY, DAAT, Cholesky and LU factorization
  ❖ Exploiting second CPU for computation-intensive kernels

■ MASSV subset
  ❖ Reciprocal, square root, reciprocal square root for single and double precision
  ❖ Single core and dual core versions

■ FFT
  ❖ Technical University of Vienna developing FFT library optimized for BlueGene/L – effective use of the SIMD FPU
BGLsim: System-Level Simulator

- Complete system-level simulator for BlueGene/L
  - Executes the full BG/L instruction set, including Hummer™
  - Models all of the devices, including Ethernet, torus, tree, lock box, and multiple levels of caches

- Efficient simulation that supports code development
  - 1,000,000 – 2,000,000 BlueGene/L instructions per second on 1GHz Pentium III
  - Timing model for single node ~10% accurate with 20% overhead

- We used it to develop/test system software
  - CNK, Linux, device drivers, networking, MPI, compilers, Math libraries, benchmarks
  - Enabled us to get system software and benchmarks running on real machine within a few days of hardware availability
IBM Research
BlueGene/L System Software Overview © 2003 IBM Corporation

512 way midplane
Full rack is ~1600lbs

BG/L Compute Rack

One of 16 node cards
One of 4 link cards
512 way midplane
Full rack is ~1600lbs

Redundant, hot plug 3-fan module
Service card
Single Node Performance Status

- **DGEMM** – 92.3% of dual core peak on 1 node
  - Observed performance at 500 MHz: 3.7 GFlops
  - Projected performance at 700 MHz: 5.2 GFlops (DGEMM performance tested in lab up to 650 MHz)

- **LINPACK**
  - 77% of peak on 1 node

- **sPPM, UMT2000** – single processor performance roughly on par with POWER3 at 375 MHz

- **FFT** – Pseudo-ops performance (5N log N) - 1.3 GFlops @ 500 MHz (TU Vienna)

- **STREAM** – impressive results even at 444 MHz
  - Tuned: Copy: 2.4 GB/s, Scale: 2.1 GB/s, Add: 1.8 GB/s, Triad: 1.9 GB/s
  - Standard: Copy: 1.2 GB/s, Scale: 1.1 GB/s, Add: 1.2 GB/s, Triad: 1.2 GB/s
  - At 700 MHz: Would beat STREAM numbers for most high end microprocessors
LINPACK Status

- Single-node – 77% of peak
- 512-node – 69% of peak
  - 1413 GFlop/s
  - Would be #35 on June 2003 TOP500 list
- Has been a great test case/driver for the hardware and system software
NAS Parallel Benchmarks – Class C on 256 nodes

- All NAS Parallel Benchmarks run successfully on 256 nodes (and many other configurations)
  - Used class C (largest class with published results)
  - No tuning / code changes
- Compared 500 MHz BG/L and 450 MHz Cray T3E
- All BG/L benchmarks were compiled with GNU and XL compilers
  - Report best result (GNU for IS)
- BG/L is a factor of two/three faster on five benchmarks (BT, FT, LU, MG, and SP), a bit slower on one (EP)
NAS Parallel Benchmark MG Class B

- 1 Cray X1 engine = 1 MSP
- 16 MSPs = ¼ cabinet (32 sq. ft.)
- 1 BlueGene/L engine = 16 nodes
- 256 nodes = ¼ cabinet (9 sq. ft.)
- Future better results in BG/L from:
  - 700 MHz clock
  - Hummer² FPU
  - Virtual node mode
ASCI Purple Benchmarks

- We are in the process of running all the ASCI Purple benchmark suite in BlueGene/L.
- Most recent activity on sPPM, UMT2K and SAGE.
- Both sPPM and UMT2K have been run on 128 nodes.
- UMT2K shows speedup of 15 when going from 8 (73.5 min) to 128 (4.9 min) nodes.
  - Includes MPI improvements as we scaled up.
- sPMM execution time ideally stays flat as number of nodes increases.
The SAGE experience

- 150,000-line Fortran90 benchmark from LANL
  - Deep understanding and modeling of the benchmark on various machines by Adolfy Hoisie’s performance team
- Satisfying outcome from porting effort during a small time window
  - LANL performance team visited Watson the week of Sept 8 - they had machine time intermittently between Wednesday-Friday
  - By the end of Thursday, SAGE was running on 8 nodes of BlueGene/L, compiled with XL (no optimizations)
  - By Friday noon, they had compiled with XL optimized (five-fold performance improvement) and had benchmarked on up to 32 nodes
  - Performance team also ran a variety of tests/microbenchmarks to verify machine behavior
- Observations from LANL team
  - BlueGene/L implements a parallel execution environment essentially free of “noise” – 3 orders of magnitude less interference on applications
  - Scalability of SAGE on small BlueGene/L similar to production ASCI machines (White and Q) with fat tree interconnects – preliminary, but encouraging
Conclusions

- We have developed a BG/L system software stack with Linux-like personality for user applications
  - Custom solution (CNK) on compute nodes for highest performance
  - Linux solution on I/O nodes for flexibility and functionality
  - MPI is the default programming model, others are being investigated

- BG/L is testing software approaches to management/operation of very large scale machines
  - Hierarchical organization for management
  - “Flat” organization for programming
  - Mixed conventional/special-purpose operating systems

- Encouraging performance results – NAS Parallel Benchmarks, ASCI Purple Benchmarks, LINPACK showing good performance

- Many challenges ahead, particularly in performance and reliability