System Software for Petascale and Beyond

Kamil Iskra
Mathematics and Computer Science Division
iskra@mcs.anl.gov
Today’s Petascale Platforms
Today’s Petascale Platforms

- **Gateway nodes**: run parallel file system client software and forward I/O operations from HPC clients.
  - 640 Quad core PowerPC
  - 450 nodes with 2 Gbytes of RAM each

- **Commodity network**: primarily carries storage traffic.
  - 900+ port 10 Gigabit Ethernet Myricom switch complex

- **Storage nodes**: run parallel file system software and manage incoming FS traffic from gateway nodes.
  - 136 two dual core Opteron servers with 8 Gbytes of RAM each

- **Enterprise storage**: controllers and large racks of disks are connected via InfiniBand or Fibre Channel.
  - 17 DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

**Bandwidths**:
- BG/P Tree 6.8 Gbit/sec
- Ethernet 10 Gbit/sec
- InfiniBand 16 Gbit/sec
- Serial ATA 3.0 Gbit/sec
Systems are Changing…

- New Constraints:
  - Transistors still scale
  - Clock leveled off (2–4 GHz)
  - Power leveled off (100–200 W)
  - ILP leveled off (2–4 ops/cycle)

- 15 years of *exponential* clock rate growth has ended

- Moore’s Law reinterpreted:
  - *Parallelism* doubles every 18 months (cores or threads)

Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith
Key Challenges

- OS kernel
- I/O infrastructure
- Parallel programming infrastructure
- Performance analysis
- Fault tolerance
- Resource management

Two main approaches towards solving them:
- Scale an existing general-purpose solution
  - familiar to future users of the system
- Develop something from scratch
  - domain-specific, complete control
RADIX Laboratory for Scalable System Software

- OS kernel: ZeptoOS
- I/O infrastructure: PVFS2, ROMIO, IOFSL, Darshan
- Parallel programming infrastructure: MPICH2
- Performance analysis: Jumpshot, Jupiter
- Fault tolerance: CIFTS
- Resource management: Cobalt, SPRUCE

- Part of Argonne's Mathematics and Computer Science Division
  - ~15 staff
  - ~5 postdocs
  - ~15 students during the summer
OS Kernel
Lightweight OS Kernels

- IBM Blue Gene: CNK
  - cycle-reproducible
  - lean (can be run under the cycle simulator)
  - no virtual memory
  - no preemption, max. 4 threads/node
  - no fork/exec

- Cray XT3: Catamount
  - (similar limitations)
  - basically abandoned by now; new XT3s come with Compute Node Linux

- Kitten
  - new open source kernel from Sandia
Linux on Compute Nodes

- Why do it?
  - features (threads, multitasking, shell scripts, Java, Python)
  - user familiarity in HPC environments
  - code portability
  - research platform
  - leverage large community of independent developers

- Key challenges:
  - jitter/noise
  - paged memory overhead
  - support for high-speed networks

http://www.zeptoos.org/
OS Jitter

- Device interrupts
- Clocktick
- Preemptive scheduling

![Graph showing OS Jitter with x-axis representing Start time (sec) and y-axis representing Duration (sec). The graph includes a red line labeled 'res-detour-zcl.dat'.]
Random detours on individual nodes delay all other nodes participating in collective operations
OS Jitter: Research Results

- Large-scale noise injection experiments:
  - longest detours most detrimental
  - short but frequent ones not really a problem
  - synchronizing detours across nodes eliminates the OS jitter problem

- Medium-scale experiments with Linux on BG/P:
  - OS jitter does not impede scalability
  - even on a vanilla kernel

- Future:
  - mainline developments in the area of tickless kernel
Memory Management

- Paged memory up to 6x slower than a static mapping
- Caused by a high cost of TLB misses on PPC450 CPUs
Memory Management: Big Memory

- Allocated at boot time
- Covered by large, semi-static TLBs
- Simple physical to virtual address mapping
Memory Management: Research Results

- Big Memory closes the performance gap
- This is not just a Blue Gene issue
  - A big memory job can get a 40-50% performance improvement if it is the very first job [after reboot, on mainstream hardware] — Don Becker, Penguin Computing

- Future:
  - short-term trends in CPUs: more flexibility (MMU in next generation BG, 1 GB pages in AMD “Barcelona”)
High Speed Networks

- Blue Gene has a high-speed 3D torus network between the compute nodes, with a DMA engine.
- The DMA engine lacks scatter/gather support required for paged memory.
- Big Memory resolves this problem by providing a physically contiguous memory region.
I/O
Compute and Storage Imbalance

*A supercomputer is a device for turning compute-bound problems into I/O-bound problems* — Seymour Cray/Ken Batcher

Current leadership-class machines supply only **1 GB/s of storage throughput for every 10 TF of compute performance**. This gap has grown by a factor of 10 in recent years.

Argonne's 557 TF Blue Gene/P (Intrepid):
- 20% of the budget spent on I/O
- Full memory dump takes over 30 minutes
  - How long does it take on your laptop?
(Sad) State of the Art…

- Typical HPC file system is a scaled up version of an enterprise product
  - Very expensive at this scale
    - Big network switch needed
  - Unsuitable API
    - Who needs POSIX locks?

- Example problems:
  - Parallel mkdir takes 10 minutes!
    - GPFS with 640 clients gives 1 mkdir/s!
  - Unaligned writes orders of magnitude slower
    - Check out the PLFS work (LANL/CMU/PSC)
  - Have you ever done “svn update” of a large repo on GPFS?

- Parallel filesystem stability/performance possibly the largest problem on contemporary large-scale systems.
Software Complexity

**High-Level I/O Library**
maps application abstractions onto storage abstractions and provides data portability.

*HDF5, Parallel netCDF, ADIOS*

**Parallel File System**
maintains logical space and provides efficient access to data.

*PVFS, PanFS, GPFS, Lustre*

---

**Application**

**High-Level I/O Library**

**I/O Middleware**

**Parallel File System**

**I/O Hardware**

**I/O Middleware**
organizes accesses from many processes, especially those using collective I/O.

*MPI-IO*
## [Part of the] Solution: I/O Forwarding

- I/O Forwarding is an additional I/O software layer for leadership-class machines that bridges the gap between application process and file systems. It reduces the number of clients seen by the file system for all applications, even without collective I/O.

<table>
<thead>
<tr>
<th>Application</th>
<th>Application with uncoordinated I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Level I/O Library</td>
<td></td>
</tr>
<tr>
<td>I/O Middleware</td>
<td></td>
</tr>
<tr>
<td>I/O Forwarding</td>
<td></td>
</tr>
<tr>
<td>Parallel File System</td>
<td></td>
</tr>
<tr>
<td>I/O Hardware</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing the layers of I/O processing and the impact of I/O Forwarding](image-url)

- Client processes visible to file system
I/O Forwarding on Blue Gene

- POSIX-only
- No aggregation, no caching
- Not extensible
ZOID: Low-Level Function Shipping Infrastructure

- **compute nodes**: syscalls, libc, FUSE, ADIO, UNIX, ZOIDFS, IP fwd, Job mgmt
- **I/O nodes**: UNIX, ZOIDFS, IP fwd, Job mgmt

- **libzoid_cn**
- **collective**
- **ZOID daemon**
The IOFSL Project

Design, build, and distribute a scalable, unified high-end computing I/O forwarding software layer that would be adopted and supported by DOE Office of Science and NNSA.

- Reduce the number of file system operations/clients that the parallel file system sees
- Provide function shipping at the file system interface level
- Offload file system functions from simple or full OS client processes to a variety of targets
- Support multiple parallel file system solutions and networks
- Integrate with MPI-IO and any hardware features designed to support efficient parallel I/O

http://www.iofsl.org/
ZOIDFS Protocol

- Opaque handles used to reference files
  - Portable across nodes
- Flexible read and write operations
  - Vectors of memory buffers and file regions
- Minimizes state to improve scalability
- Reduces the number of I/O operations
- Enables middleware optimizations
- Example call:
  ```c
  int zoidfs_read(const zoidfs_handle_t *handle,
                   zoidfs_size_t mem_count,
                   void *mem_starts[],
                   const zoidfs_size_t mem_sizes[],
                   zoidfs_size_t file_count,
                   const zoidfs_ofs_t file_starts[],
                   zoidfs_size_t file_sizes[]);
  ```
IOFSL: Performance Optimizations

- Reduced number of metadata operations
  - Lookup a handle from one process, broadcast to others via MPI
- Reduced number of file data operations
  - Complex datatypes can be handled with a single call
- Pipelining
  - Large I/O operations exposed to the forwarding server
  - Simultaneous transfer of data between CN-ION and ION-FS
- Aggregation
  - Reduce the number of requests
- Caching of file data and metadata
Results
**ZeptoOS Matches the Performance of CNK...**

- **NAS Parallel Benchmarks (class C / 1024 nodes)**

<table>
<thead>
<tr>
<th></th>
<th>CNK (Mop/s)</th>
<th>Zepto (Mop/s)</th>
<th>Zepto/CNK</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>3990.92</td>
<td>4009.76</td>
<td>1.005</td>
</tr>
<tr>
<td>CG</td>
<td>15749.35</td>
<td>15706.83</td>
<td>0.997</td>
</tr>
<tr>
<td>MG</td>
<td>134955.02</td>
<td>134380.19</td>
<td>0.996</td>
</tr>
<tr>
<td>FT</td>
<td>96594.32</td>
<td>96385.49</td>
<td>0.998</td>
</tr>
<tr>
<td>LU</td>
<td>40889.58</td>
<td>40616.70</td>
<td>0.993</td>
</tr>
<tr>
<td>EP</td>
<td>2503.11</td>
<td>2499.84</td>
<td>0.999</td>
</tr>
<tr>
<td>SP</td>
<td>106009.42</td>
<td>105708.94</td>
<td>0.997</td>
</tr>
<tr>
<td>BT</td>
<td>165240.30</td>
<td>164777.07</td>
<td>0.997</td>
</tr>
</tbody>
</table>

- **NAS Parallel Benchmarks FT (class D)**

<table>
<thead>
<tr>
<th></th>
<th>CNK (Mop/s)</th>
<th>Zepto (Mop/s)</th>
<th>Zepto/CNK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>217666.44</td>
<td>216916.86</td>
<td>0.997</td>
</tr>
<tr>
<td>4096</td>
<td>371444.68</td>
<td>372516.63</td>
<td>1.003</td>
</tr>
<tr>
<td>8192</td>
<td>768919.56</td>
<td>768431.17</td>
<td>0.999</td>
</tr>
</tbody>
</table>
... And Sometimes Exceeds It

- **Parallel Ocean Program**

<table>
<thead>
<tr>
<th></th>
<th>CNK (s)</th>
<th>Zepto (s)</th>
<th>CNK/Zepto</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>196.62</td>
<td>197.26</td>
<td>0.997</td>
</tr>
<tr>
<td>128</td>
<td>105.69</td>
<td>105.59</td>
<td>1.001</td>
</tr>
<tr>
<td>256</td>
<td>57.37</td>
<td>57</td>
<td>1.006</td>
</tr>
<tr>
<td>512</td>
<td>34.98</td>
<td>34.49</td>
<td>1.014</td>
</tr>
<tr>
<td>1024</td>
<td>22.37</td>
<td>21.89</td>
<td>1.022</td>
</tr>
<tr>
<td>2048</td>
<td>16.74</td>
<td>16.32</td>
<td>1.026</td>
</tr>
<tr>
<td>4096</td>
<td>14.54</td>
<td>14.10</td>
<td>1.031</td>
</tr>
</tbody>
</table>

- Caused by `gettimeofday()` being 7x more expensive under CNK
LOFAR

LOw Frequency Array

- revolutionary radio telescope
  - no dishes
  - $\mathcal{O}(10000)$ receivers
  - omni-directional
- central processing
  - real time
  - software
  - BG/P supercomputer
LOFAR BG/L Processing with ZOID

- reorder, filter, correlate data
- use ZOID plug-in on I/O node

- application on I/O node: no need for input cluster
Falkon: Managing 160,000 CPUs

- Slower shared storage
- High-speed local disk

Diagram showing Falkon and its connections to different storage types.
DOCK on BG/P: ~1M Tasks on 118,000 CPUs

- CPU cores: 118784
- Tasks: 934803
- Elapsed time: 7257 sec
- Compute time: 21.43 CPU years
- Average task time: 667 sec
- Relative Efficiency: 99.7%
  - (from 16 to 32 racks)
- Utilization:
  - Sustained: 99.6%
  - Overall: 78.3%

- GPFS
  - 1 script (~5KB)
  - 2 file read (~10KB)
  - 1 file write (~10KB)

- RAM (cached from GPFS on first task per node)
  - 1 binary (~7MB)
  - static input data (~45MB)
Other Issues
Fault tolerance: CIFTS

Coordinated Infrastructure for Fault Tolerant Systems

- Traditional fault tolerance is handled by individual components
- No coordination between them
- No sharing of fault information
- Components don't know the reason for system-wide faults
  - Did the application exit due to an inherent error in the code?
  - Did it exit due to a system failure?

http://www.mcs.anl.gov/research/cifts/
Fault Tolerance Backplane:

- Provides a scalable framework to exchange fault-related information
- Exposes a standard interface that can be used by any component
- Provides a uniform event handling and notification mechanism
Performance Analysis: Jupiter

Visual Characterization of I/O System Behavior for High-End Computing

- Plenty of research on application performance analysis and debugging tools
- The needs of system software developers often overlooked
  - a high-scale parallel filesystem is a complex parallel application
- Develop/improve/deploy:
  - end-to-end, scalable tracing integrated into the I/O system (MPI-IO, I/O forwarding, file systems),
  - new visual representations and analysis techniques for inspecting traces and extracting knowledge, scalable to very large systems and integrable with existing techniques
Jupiter
Conclusion
What’s Next?

- All Large Core
- Mixed Large and Small Core
- Many Small Cores
- All Small Core
- Many Floating-Point Cores
- + 3D Stacked Memory

Source: Jack Dongarra, ISC 2008
The Big Questions

- Extreme-scale operating systems will be even more challenging on emerging next-generation hardware
  - Large multi-core
  - Heterogeneous cores
  - Hierarchical

- What should the OS stack look like?
  - virtualization/partitioning?
    - For example: should we partition OS services to $n-1^{\text{th}}$ core?
  - I/O forwarding inside each compute node?
Could We Get Rid of Enterprise Storage in HPC?

- What if we collapse I/O forwarding nodes and file server nodes?

- What is the difference?
  - no external network switch
  - high-speed HPC network used instead
Collaborators

- RADIX: Kazutomo Yoshii, Harish Naik, Pete Beckman, Dries Kimpe, Jason Cope, Rob Ross, Phil Carns, Sam Lang, Rob Latham, Rinku Gupta, Rusty Lusk
- Project partners:
  - University of Oregon: Allen Malony, Sameer Shende, Aroon Nataraj, Alan Morris
  - Los Alamos: James Nunez, John Bent, Gary Grider, Sean Blanchard, Latchesar Ionkov, Hugh Greenberg
  - Oak Ridge: Steve Poole, Terry Jones
  - Sandia: Lee Ward
  - UC Davis: Kwan-Liu Ma, Chris Muelder
- External collaborators:
  - ASTRON (Netherlands Institute for Radio Astronomy): John W. Romein, P. Chris Broekema
  - University of Chicago: Michael Wilde, Zhao Zhang, Ioan Raicu, Allan Espinoza
  - University of Delaware: Guang R. Gao, Handong Ye
- Summer students: Nawab Ali, Ivan Beschastnikh, Peter Boonstoppel, Hajime Fujita, Valerie Galluzzi, Jason Kotenko, Alex Nagelberg, Kazuki Ohta, Satya Popuri, Taku Shimosawa, Zichen Xu, Kazunori Yamamoto