Summit

A Next-Generation Supercomputer for CU-Boulder

Thomas Hauser
thomas.hauser@colorado.edu

Peter Ruprecht
peter.ruprecht@colorado.edu

www.rc.colorado.edu

Outline

- Overview of project and general architecture
- More details about types of compute nodes
- Other hardware: Omni-Path interconnect and GPFS scratch filesystem
- Differences with Janus
- Getting the most out of Summit
- Installation and availability schedule
- Own a part of Summit
- Questions and discussion

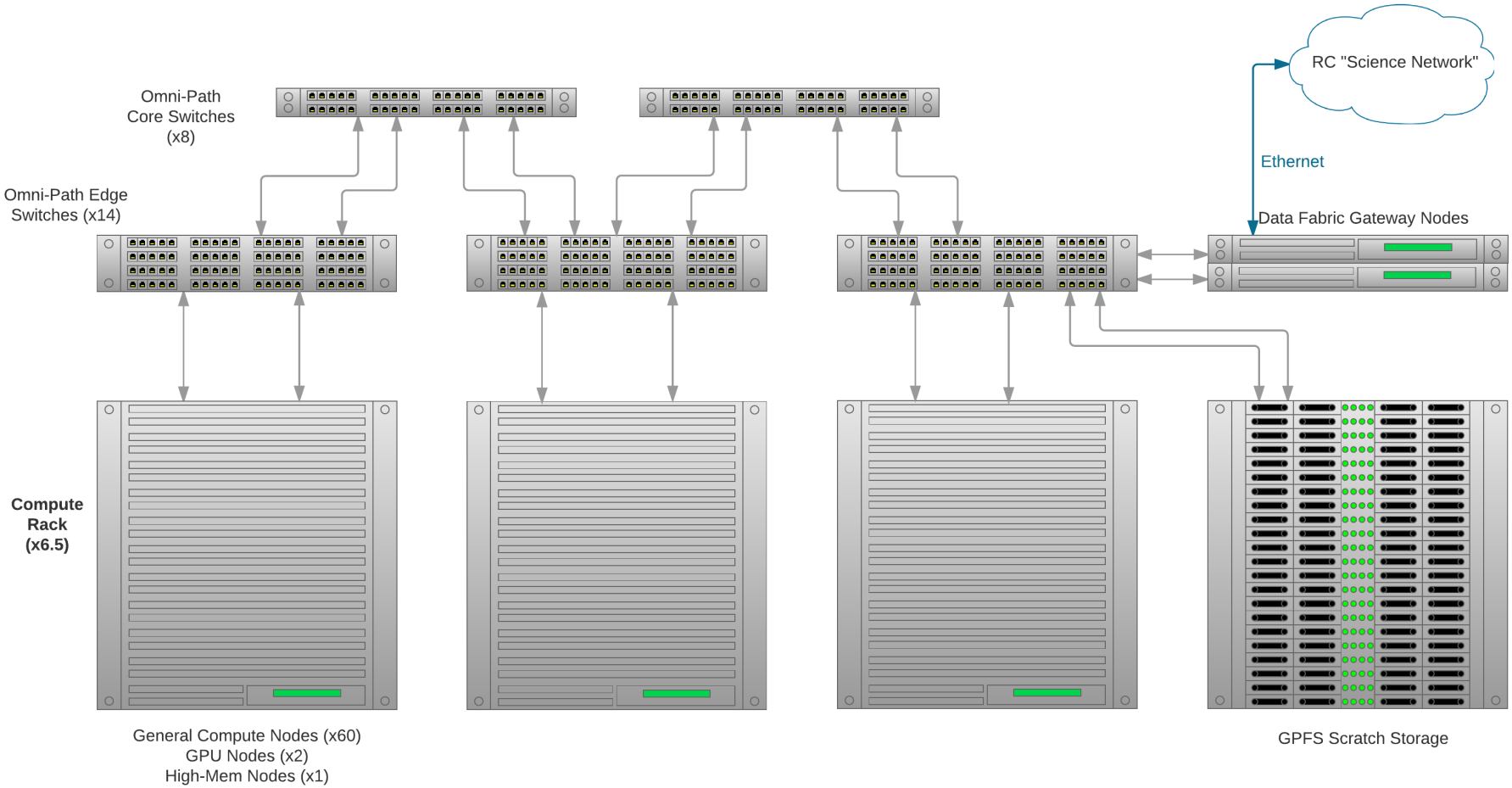
Summit: CU-Boulder's Next-Generation Supercomputer

- Funded via an NSF MRI grant awarded jointly to CU-Boulder and CSU
- Installed and running by fall semester 2016
- *About 67% of core-hours available to CU-Boulder*
- Theoretical peak performance around 450 TFLOPS* (compared with about 170 for Janus)
- Mix of established and cutting-edge technology
- Expected 5-year useful lifetime
- Principal vendor and integrator is Dell

*trillion floating point operations per second

SUMMIT SCHEMATIC

Peter Ruprecht | July 15, 2016



Summit: Node Types

- 380 general compute nodes
  - 24 real cores and 128 GB RAM
  - local SSD
- 10 GPGPU/visualization nodes
  - 2x NVIDIA K80 GPUs
- 5 High-memory nodes
  - 2 TB RAM each
- 20 Xeon Phi (“Knights Landing”) nodes
  - Phi installed directly in CPU socket; not a separate card
  - 72 cores / 288 threads “many core”
  - installed in second phase, late 2016

Dell C6320 Compute Node

Image: Dell.com

Omni-Path (OPA) Interconnect

- Cutting-edge network product from Intel
- Same role as the InfiniBand fabric on Janus
  - Also carries NFS traffic from /home, /projects, /work ...
- 100 Gb/s bandwidth
- Extremely low latency for MPI performance
- Fat-Tree topology with 8 core switches and 32 nodes per edge switch (easily expandable)
- "Islands" of 32 nodes fully non-blocking
- 2:1 blocking factor between islands

GPFS Scratch Storage

- 1 PB of high-performance scratch storage
- GPFS for parallel access and improved small-file performance
- Directly-attached to OPA fabric (in a later update)
- Expandable to about 2 PB; performance also scales up as more drives are added
- DataDirect Networks SFA14K "GridScaler" appliance with embedded and external file servers
- Expect >20 GB/s parallel throughput and >10K file creations per second

GPFS Scratch Storage

- Available on login nodes and data-transfer nodes, but not on non-Summit compute nodes
- Mounted as /scratch/summit
- We'll provide a way to copy data from Janus Scratch

Summit vs Janus

Feature	Janus	Summit
CPU cores/node Clock frequency	12 2.8 - 3.2 GHz	24 2.5 - 3.3 GHz
RAM/core Memory bandwidth	2 GB 32 GB/s	5 GB 68 GB/s
Interconnect type Bandwidth Latency	QDR InfiniBand 40 Gb/s 1.2 us	Omni-Path 100 Gb/s 0.4 us
Filesystem	Lustre – optimized for large parallel transfers	GPFS – good at parallel transfers; also good at small file operations
Vectorization	SSE4 (4 operations/cycle)	AVX2 (16 operations/cycle)

Why Optimize for Summit?

- Summit has fewer nodes than Janus, thus we can allocate fewer core-hours (SU)
- Summit is shared with CSU and RMACC, thus we can allocate fewer core-hours to CU-Boulder

- If you do not take advantage of Summit's performance features, your allocation will not go as far as it could, so
  - You will get fewer results
  - You will publish fewer papers
  - You will take extra years to graduate
  - You will not get that awesome faculty position

Getting the most out of Summit

- Any application that you have built for Janus will need to be recompiled
- Applications ideally should take advantage of multi-core / many-core architectures (ie, parallelize)
- Performance improvements in latest processors are mainly through more cores and SIMD* rather than faster clock speed
- Applications should be parallelized and “vectorized” … otherwise Summit may *seem* slower than Janus

*single instruction on multiple data

Vectorization overview

- Vectorizing a computation lets it take advantage of SIMD instruction sets
- That is, during one CPU cycle a single CPU core can do the same operation on several data objects (16 at once in the case of Summit)
- The compiler can auto-vectorize some loops; the programmer needs to write code that the compiler can easily vectorize
- Some math libraries, such as Intel MKL, are fully optimized for vectorization
- Many applications have vectorization built in
  - Matlab
  - Python provided by RC

More on Vectorization

Data register:

data 1	data 2	data 3	data 4

Consider this loop:

```
for (i=1 ; i<=N ; i++)
    b[i] = 3 + a[i];
```

(inspired by Intel Autovectorization Guide)

Without vectorization:

3 + a[1]	not used	not used	not used

With vectorization:

3 + a[1]	3 + a[2]	3 + a[3]	3 + a[4]

Enabling Vectorization

- Set appropriate compiler flag
  - `-march=core-avx2`
  - `-vec-report` or `-qopt-report`
- Loop requirements:
  - Exit from loop must not be data-dependent, i.e. loop trip count must be known at runtime
  - No branching within the loop (`if`-statements allowed in some cases)
  - No function calls within the loop (except intrinsic math functions that are vectorized)
- Align data on register boundaries
- Be careful with C pointers (try `-restrict` flag)
- Give the compiler hints whenever possible

I/O Optimization

- Total Job Time = Computational Time
  + Communication Time + I/O time
- Thus, optimizing and parallelizing I/O can be important
- Parallel access requires special programming constructs or libraries, such as MPI-IO
  - MPI-IO consists of MPI functions for data reading and writing
  - Each MPI process can read or write a portion of a shared file
- Structured data formats such as HDF5 can help a lot

More on I/O Optimization

- Put no more than 10K files in a single directory
  - Use a structure of subdirectories for holding >10K files
- Avoid metadata intensive operations like `ls -l`
- When programming:
  - If a file only needs to be read, open it read-only
  - Do not open and close files too frequently
  - Assign a subset of cores for handing I/O
  - Give MPI-IO “hints” about how your I/O operations should be tuned

Other New Features w/Summit

- OS is RedHat Enterprise Linux 7 (vs RHEL6 on Janus)
- RC login nodes will also be updated to RHEL7
- Slurm job scheduler will have some new features
  - Different queue names
  - Expectation for requesting node features
  - Allowing shared nodes, for single-core jobs
- Allocations will be provided as shares rather than “bank accounts”; you’ll get higher queue priority if you have a larger share
- Running on different node types will “cost” different amounts

Schedule

- Late April – mini-Summit test cluster available for software builds and small-scale testing
- Mid July – Summit delivered and installed in HPCF
- Late July – initial acceptance testing
- Early Aug – application testing and early user access
- Early Oct – general availability
- Dec-Jan – Knights Landing Phi nodes installed; possible expansion of general compute

Want early access?

- Benefits to being a beta tester
  - Your code and workflow is ready once Summit is deployed
  - RC staff will help you to make sure your code works well on Summit
  - Get a computational project done during the initial phase of Summit when it might be less busy
- Requirements to become a tester
  - Have a well-understood code ready for deployment
  - We are looking for a variety of applications from different fields, including GPU-enabled apps
  - Tolerant to unexpected problems or changes
    - The high-speed network on Summit is “bleeding edge” and we may have to change things or things may not work as expected

Buying into Summit

- You can own additional nodes
  - $8840 for a general compute node + scratch storage
    - $18000 for a GPU node
    - $38950 for a high-mem node
    - CU-Boulder and CSU are offering a $2000/node subsidy
  - Purchase mechanism for non-CU/CSU is still being worked out
  - Priority access to an allocation sized to what your node(s) would provide you (210K core-hours/yr for a general compute node)
  - Additional purchases need to be finalized by Nov 2016

Thank you! Questions?

CU-Boulder Research Computing

www.rc.colorado.edu rc-help@colorado.edu

Slides:

https://github.com/ResearchComputing/Final_Tutorials/

Link to tutorial feedback survey:

http://goo.gl/forms/8VidcwOhRT

Thank you! Questions?

CU-Boulder Research Computing

www.rc.colorado.edu rc-help@colorado.edu