Application Scalability and Parallel I/O

William Gropp
University of Illinois at Urbana-Champaign
www.cs.illinois.edu/~wgropp

1867

What are the Problems?

- Performance problems aren't always where you think
  - ♦ Load and compute resource imbalance can show up as slow communication
  - ♦ I/O performance and reliability sensitive to access patterns, configuration

Load Imbalance and Scalability

- Sources of Imbalance
  - ♦ OS and Runtime share cores, memory, network
  - ♦ Application shares network, I/O system
  - ♦ Few applications have exactly perfect load balance
- Tools already available to explore
  - ♦ Can customize tools such as FPMPI to provide application-specific information; correlate with node "noise"
  - ♦ New MPI_T interface can provide additional details

Improving Load Balance and Core Use

- Use a hybrid (MPI+OpenMP or MPI+OpenACC) approach to simplify shifting work between cores
  - ♦ Have developed new loop schedulers that provide better data locality, lower overhead. See poster for details
- Appropriate for applications already using a hybrid model or planning to adopt soon

Improving Load Balance and Core Use

- Use MPI-3 shared memory "MPI+MPI"
  - ♦ New with MPI-3, supported on Blue Waters
  - ♦ Allows MPI processes on the same node (or chip) to allocate memory that is shared between those processes
    - Access to memory is through language, e.g., a[72]=2; rather than MPI
  - ♦ Since all MPI processes share the memory, they can all easily redistribute work

Improving Load Balance and Core Use

- Improved graph and workload partitioning
  - ♦ Many codes use a graph partitioner to load balance work among MPI processes
  - ♦ Good code exist, but
    - All are based on a cost model for nodes, edge cuts
- Cost models often too inaccurate
  - ♦ Ignore network contention, core/chip/node placement, overly simple communication cost, impact of partition on computation cost
  - ♦ Some parts impossible to do at partition time
    - Mapping onto physical hardware, impact of other jobs
- Approach: consider iterative refinement of partition based on measurements

Parallel I/O Performance

- I/O performance for the same data operation can vary
  - Example: 1024 processes, write 16kx16k array to a single file. Note only 64 nodes.

Stripes	Stripe Unit	Bandwidth MB/sec	Collective I/O?
1	Default	2.87	No
16	16MB	15.5	No
1	Default	371	Yes
16	16MB	3,850	Yes

Parallel I/O Performance

- Currently collecting data on use with Darshan
  - ♦ Over 70k runs already
  - ♦ Will examine to look for potential opportunities
- No easy recipe
  - ♦ Luu et al (HPDC'15) have shown that common I/O patterns can provide either good or awful performance, depending on details
  - ♦ Fixes need collaboration with teams
    - Everything from setting environment variables or using MPI_Info on file open to code restructure to use alternative I/O patterns

Parallel I/O Performance

- Approach
  - ♦ Use Darshan data to identify potential for I/O performance improvement
  - ♦ May develop application-customized profiling tools to discover details
- Performance enhancement techniques
  - ♦ Tune I/O parameters (use autotuning)
  - ♦ Enable or recode to use buffered I/O
  - ♦ Restructure to use collective I/O, adapt to application workflow

1867

Summary

- Performance can be lost anywhere
- Rules of thumb may be misleading
- Changes for load balance, I/O will apply to most systems
- Specifics depend on the application. Come see the poster for more information!