CMSC 714
Lecture 24
Data Intensive Applications

Alan Sussman

Notes

- Group Project presentations next Tuesday (and Thursday if needed)
  - final report due Monday, December 12
- Zaratan opening celebration on Thursday in Stamp Atrium from 1-3PM on Thursday
  - You should all have received invitations
- Course evaluation (Student Feedback on Course Experiences) web site open
  - https://www.CourseEvalUM.umd.edu

Virtual Microscope

- Software emulation of a light microscope, to view and manipulate very large slide images, built at UMD and Johns Hopkins med school
  - for viewing and processing images captured from standard pathology specimens (need special purpose hardware for high throughput data capture)
  - problem is very large data sizes
    - a slide is maybe 30K pixels on a side at high resolution, so one focal plane is maybe 10GB uncompressed
    - and need multiple focal planes for some samples
    - and JHU hospital produces >400K slides per year (in 1998!)
- Client/server system design
  - client runs on user desktop machine – Java GUI
  - server stores, retrieves, processes slide image data on parallel machine or workstation cluster
    - implemented both with Active Data Repository OO framework and with DataCutter component framework

Virtual Microscope

- Client provides drag/zoom interface to browse through a slide
  - use thumbnail to keep track of where you are on a slide
  - standalone client can cache image data for improved response time – using both memory and disk on client machine
- Server basic computation is map-reduce
  - map one or more input pixels at highest resolution to desired output resolution, and aggregate if multiple pixels map to same output pixel
- Active Data Repository
  - user defined functions used for map and reduce, framework orchestrates parallel execution across data stored on multiple nodes of a cluster or parallel machine
  - data blocks distributed across disks for parallel access and are indexed for fast retrieval (more important for more complex map functions)
  - images also need to be decompressed from stored JPEG form before map and reduce steps, and clipped to query window
  - experiments show that ADR implementation scales well, to handle multiple clients, with low overhead

Virtual Microscope

- DataCutter
  - component framework for processing large datasets in a distributed environment
  - filter-stream programming model
    - each filter is a component, and filters connected via streams, which deliver data buffers between filters
  - supports flexible placement of filters, filter replication for load balancing (transparent copies)
  - VM filter pipeline is: read-data, decompress, clip, zoom, view
- Performance results show that DataCutter implementation deals better than ADR with load balance issues, but ADR can process large queries faster from parallel execution of a single query
  - for DataCutter, filter placement matters – communication between filters adds latency if on different hosts
- Overall performance results for VM show that can achieve interactive response times for real slide data, on not-too-large server system configurations

Global Weather Simulations

- Combine high performance high-resolution ensemble weather simulation with data assimilation
  - A *coupled* application
- And run on Fugaku, the number 2 and previous number 1 machine on the TOP500
  - An ARM CPU-only system, so targeted at a wide range of applications, with vector extensions
  - Hierarchical storage system, with SSDs (one I/O node for every 16 compute nodes for high bandwidth) and a Lustre parallel file system for capacity
- Bottom line is that the application scales (weakly) to the full system (but results mainly shown on part of the system), and does a lot of I/O (petabytes per cycle)

NICAM-LETKF

- NICAM is a global cloud-resolving weather prediction model
  - And can get to high spatial high resolution, with results shown down to 3.5km resolution
  - Overall application performance is measured in SDPH (simulation days per hour of wall clock time), and related work has shown performance of similar models at a fraction of SDPH (.1-.7 maybe) on large CPU and GPU systems
- Data assimilation via LETKF (local ensemble transform Kalman filter) is a method for incorporating observation data to get better initial conditions for each round of simulation
- The NICAM ensemble simulations with up to 1024 members run for 9 hours (of simulated time) in a *cycle*, producing output every hour after 3 hours, and the output at 6 hours is used in LETKF along with observation data to produce input conditions for the next cycle

Optimizations

- 75% of the time in a DA cycle is in NICAM
  - But the bottlenecks are spread throughout the code, so lots of separate optimizations needed to improve computation and data movement
  - Use both SIMD instructions and HBM with an overall goal of minimizing bytes/flop
  - Use single precision FP for both speed and decrease data movement and I/O requirements
    - Ensemble computations help with overall accuracy
- Some opts for LETKF too
  - Big one is eigensolver optimized for small real symmetric square matrices, based on the one for the K computer
    - Uses a special matrix layout too, for better use of cache

Performance

- Amount of I/O is very large
  - NICAM outputs 1.4PB per cycle, with 104TB used for next cycle of simulation
  - Separate files per process on SSD for best performance
- Performance shown for grid sizes from 56km down to 3.5km, both in double and mixed precision, on up to the full Fugaku system (131K nodes, 524K PEs) for the DA runs
- Mixed precision runs maybe 1.6x faster than double per cycle for NICAM, and DA runs also much faster in mixed precision and scale better
- Overall performance is impressive
  - 23 PFLOPs for NICAM on 3.5km grid, mixed precision, 256 member ensemble run on 131K nodes (5.3% of peak performance)
  - LETKF does not scale as well, because of all-to-all communication in input data redistribution, with results for a 14km grid with mixed precision on 8K nodes and 1/16 data size running at 0.34 SDPH and about 5.5PFLOPs
  - When run together on 3.5km grid, with only an estimated NICAM part (1 hour of simulated time), get about 0.1 SDPH, 29PFLOPs for NICAM and 79PFLOPs for DA, but around 80% of total time spent in NICAM