SPEEDING UP PARALLEL SIMULATION WITH AUTOMATIC LOAD BALANCING

Hari Govind, Gengbin Zheng, Laxmikant Kale, Michael Breitenfeld, Philippe Geubelle
University of Illinois at Urbana-Champaign

Motivations
- Parallel machines abound
- Capabilities enhanced as machines get more powerful
- PSC Lemieux, ASCI White, Earth Simulator, BG/L
- Clusters becoming ubiquitous
- Desktops and Games consoles go parallel
- Cell processor, multi-core chips
- Applications get more ambitious and complex
- Adaptive algorithms
- Irregular or dynamic behavior
- Multi-component and multi-physics
- MPI based code limitations
- No adaptive load balancing

Applications
- CSE applications
- Crack propagation
- Adaptive mesh refinement
- Molecular dynamics
- NAMD
- Cosmology simulation
- Fault tolerance

Processor Virtualization

Programmer: [Over] decomposition into virtual processors (VP)
Runtime: Assign VPs to processors
Enables adaptive runtime strategies

AMPI: Adaptive MPI
- Each virtual process implemented as a user-level thread embedded in a Charm++ object

Benefits
- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Computation performed upon receipt of a message
- Adaptive overhead of communication
  - Predictability:
    - Automatic out-of-core execution
    - Asynchronous reductions
- Dynamic mapping
  - Heterogeneous clusters
  - Vacate, adjust to speed, share
  - Automatic checkpointing/restarting
  - Automatic dynamic load balancing
  - Change set of processors used
  - Communication optimization

How to Migrate Objects
- Objects
  - Packing/unpacking functions
- User-level Threads
  - Global variables:
    - ELF object format: switch GoT pointer
    - Alternative: compiler/pre-processor support
  - Migration of stack:
    - Isomalloc (from PM2 in France):
    - Reserve virtual space on all processors for each thread
    - Mmap it when you migrate there
  - Migration of Heap data:
    - Isomalloc:
    - User-supplied or compiler generated pack function

Charm++ Architecture

Processor A’s Memory

Processor B’s Memory

Real Processors

Load Balancing Strategies

Parallel Framework for Unstructured Meshes (ParFUM)

Seed Load Balancing
- Tasks are initially represented by object creation messages, or "seeds"
- Seed load balancing involves the movement of seeds, to balance work across processors
- Low responsiveness
- Load balancing request blocked by long entries
- Neighborhood averaging with work-stealing when filling immediately
- Interruption-based message
- Fast response to the request
- Work-stealing at idle time

Measurement Based Load Balancing
- Based on Principle of persistence
- Runtime instrumentation
  - Measures communication volume and computation time
  - Measurement based load balancers
- Use the instrumented data-base periodically to make new decisions
- Many alternative strategies can use the database
  - Centralized vs. distributed
  - Greedy improvements vs. complete reassignments
- Taking communication into account
- Taking dependencies into account (More complex)
- Topology-aware

Principle of Persistence
- Once an application is expressed in terms of interacting objects, object communication patterns and computational loads tend to persist over time
  - In spite of dynamic behavior
  - Abrupt and large but infrequent changes (eg: AMR)
  - Slow and small changes (eg: particle migration)
- Parallel analog of principle of locality
- Heuristics, that holds for most CSE applications

Parallel Programming Laboratory

http://charm.cs.uiuc.edu
SPEEDING UP PARALLEL SIMULATION WITH AUTOMATIC LOAD BALANCING
Hari Govind, Gengbin Zheng, Laxmikant Kale, Michael Breitenfeld, Philippe Geubelle
University of Illinois at Urbana-Champaign

Crack Propagation Simulation
- 1-D elastic-plastic wave propagation
- crack propagates in an elastic wave
- bar is dynamically loaded
- bar, upon reflection
- dynamic wave
- fracture propagates
- 1-D elastic-plastic wave
- plastic zone

Molecular Dynamics Simulation
- Molecular dynamics and related algorithms
- e.g., minimization, steering, locally enhanced sampling
- efficient algorithms for full electrostatics
- effective on affordable commodity hardware
- building a complete modeling environment
- written in Charm++

Load Balancing on Very Large Machines
- Scalability limits
- consider an application with 1M objects on 64K processors
- metrics for a multi-dimensional optimization
- memory usage on any one processor
- decision making time
- quality of load balancing decision

Load Balancing in Fault Tolerance
- double in-memory checkpoint/restart
- does not rely on extra processors
- maintains execution efficiency after restart

Future work
- apply adaptive load balancing framework for increasingly complex simulations
- adaptive insertion/activation of cohesive elements for dynamic fracture simulations
- adaptive mesh adaptation
- conduct experiments using the load balancing framework on very large parallel machines such as Blue Gene/L
- requires mesh to be partitioned into very large number of chunks
- experiment with the hierarchical load balancing strategy

Parallel Programming Laboratory
http://charm.cs.uiuc.edu