The Asynchronous Dynamic Load-Balancing Library

Rusty Lusk, Steve Pieper, Ralph Butler, Anthony Chan

Mathematics and Computer Science Division
Nuclear Physics Division
Argonne National Laboratory
Outline

- Reminders about ADLB
  - What it is (PF 2007)
  - How to use it (PF 2008)
- This year: how it works
- Recent progress
- Challenges remaining
Master/Slave Algorithms and Load Balancing

- Advantages
  - Automatic load balancing

- Disadvantages
  - Scalability - master can become bottleneck

- Wrinkles
  - Slaves may create new work
  - Multiple work types and priorities that impose work flow
The ADLB Vision

- No explicit master for load balancing; slaves make calls to ADLB library; those subroutines access local and remote data structures (remote ones via MPI).
- Simple Put/Get interface from application code to distributed work queue hides most MPI calls
  - Advantage: multiple applications may benefit
  - Wrinkle: variable-size work units, in Fortran, introduce some complexity in memory management
- Proactive load balancing in background
  - Advantage: application never delayed by search for work from other slaves
  - Wrinkle: scalable work-stealing algorithms not obvious
The ADLB Model (no master)

- Doesn’t really change algorithms in slaves
- Not a new idea (e.g. Linda)
- But need scalable, portable, distributed implementation of shared work queue
  - MPI complexity hidden here.
API for a Simple Programming Model

- Basic calls
  - ADLB_Init( num_servers, am_server, app_comm)
  - ADLB_Server()
  - ADLB_Put( type, priority, len, buf, answer_dest )
  - ADLB_Reserve( req_types, handle, len, type, prio, answer_dest)
  - ADLB_Ireserve( … )
  - ADLB_Get_Reserved( handle, buffer )
  - ADLB_Set_Done()
  - ADLB_Finalize()

- A few others, for tuning and debugging
  - ADLB_{Begin,End}_Batch_Put()
  - Getting performance statistics with ADLB_Get_info(key)
Parallel Sudoku Solver with ADLB

Program:

if (rank = 0)
    ADLB_Put initial board
    ADLB_Get board (Reserve+Get)
while success  (else done)
    ooh
    find first blank square
    if failure  (problem solved!)
        print solution
        ADLB_Set_Done
    else
        for each valid value
            set blank square to value
            ADLB_Put new board
        ADLB_Get board
    end while

Work unit =
    partially completed “board”
**How it Works**

- After initial Put, all processes execute same loop (no master)
Optimizing Within the ADLB Framework

- Can embed smarter strategies in this algorithm
  - ooh = “optional optimization here”, to fill in more squares
  - Even so, potentially a lot of work units for ADLB to manage

- Can use priorities to address this problem
  - On ADLB_Put, set priority to the number of filled squares
  - This will guide depth-first search while ensuring that there is enough work to go around
    - How one would do it sequentially

- Exhaustion automatically detected by ADLB (e.g., proof that there is only one solution, or the case of an invalid input board)
Experiments with GFMC/ADLB on BG/P

- Using GFMC to compute the binding energy of 14 neutrons in an artificial well ("neutron drop" = teeny-weeny neutron star)
- A weak scaling experiment

<table>
<thead>
<tr>
<th>BG/P cores</th>
<th>ADLB Servers</th>
<th>Configs</th>
<th>Time (min.)</th>
<th>Efficiency (incl. serv.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K</td>
<td>130</td>
<td>20</td>
<td>38.1</td>
<td>93.8%</td>
</tr>
<tr>
<td>8K</td>
<td>230</td>
<td>40</td>
<td>38.2</td>
<td>93.7%</td>
</tr>
<tr>
<td>16K</td>
<td>455</td>
<td>80</td>
<td>39.6</td>
<td>89.8%</td>
</tr>
<tr>
<td>32K</td>
<td>905</td>
<td>160</td>
<td>44.2</td>
<td>80.4%</td>
</tr>
</tbody>
</table>

- Recent work: “micro-parallelization” needed for $^{12}\text{C}$, OpenMP in GFMC.
How It Works

- Real numbers: 1000 servers out of 32,000 processors on BG/P
  - And recently introduced other communication paths
The ADLB Server Logic

Main loop:
- MPI_Iprobe for message in busy loop (emit diagnostics)
- MPI_Recv message
- Process according to type (20 types)
  - Update status vector of work stored on remote servers
  - Manage work queue and request queue
  - (may involve posting MPI_Isends to isend queue)
- MPI_Test all requests in isend queue
- Return to top of loop

The status vector replaces single master or shared memory
- Circulates every .1 second at high priority
ADLB Uses Multiple MPI Features

- ADLB_Init returns separate application communicator, so application can use MPI for its own purposes if it needs to.
- Servers are in MPI_Iprobe loop for responsiveness.
- MPI_Datatypes for some complex, structured messages (status)
- Servers use nonblocking sends and receives, maintain queue of active MPI_Request objects.
- Queue is traversed and each request kicked with MPI_Test each time through loop; could use MPI_Testany.
- Client side uses MPI_Ssend to implement ADLB_Put in order to conserve memory on servers, MPI_Send for other actions.
- Servers respond to requests with MPI_Rsend since MPI_Irecvs are known to be posted by clients before requests.
- MPI provides portability: laptop, Linux cluster, SiCortex, BG/P
- MPI profiling library is used to understand application/ADLB behavior.
Looking at GFMC/ADLB with Jumpshot (in the good old days)
Things Can Get Worse at Larger Scale
Experiments Last Fall
Experiments Last Fall
**Good News – Bad News**

**RESULTS SO FAR**

ADLB performance is very good up to 8192 nodes (32,768 cores)

![Efficiency graph showing ADLB and ADLB+OMP performance over time]

Efficiency = compute_time/wall_time – 9 Jun 2009

- ADLB
- Jun 2009
- ADLB+OMP
- Feb 2009
- $^{12}$C

Number of nodes (4 OMP processes per node)
The Need for Tools

- **Understanding** the behavior of the coupled application/library is difficult.
  - (Friendly) finger pointing has led to advances
- Big problem: everything works fine at 8,000 processors and below
  - So testing and debugging is cumbersome at best
- Jumpshot not really usable at very large scale
- Statistics point to problems, but not to solutions, since time-varying behavior is not captured in averages
- Large amounts of debugging and monitoring output cause their own problems
- We are still developing tools for understanding behavior
  - At large scale
  - That varies over time
Plotting Statistics Over Time

$^{12}$C – slow $\Psi_T$ – 1024 nodes – 86.6% efficiency – 1 Jun 09
Tracking Anomalies

$^{12}$C – 16,384 BGP nodes – ADLBm383-t1 – 6 Jun 09

Undirected work packages from 0 to 22 minutes; total reserve min = 87638
Problem Apparently Fixed

$^{12}$C – 16,384 BGP nodes – ADLBm383-t1 – 6 Jun 09

Reserve time (seconds)

Dwell time (seconds)

204
0
2972
260
617
47

All packages
Multiple Load-Balancing Regimes

- The original objective was to do balancing of processing load
- Much of the last year has been spent on balancing of the memory load
  - Work units may be moved from server to server
  - Even proactively
- We may now be having problems that can only be solved by balancing of the message-passing load.
The “Official” Questions

What are the main accomplishments since the last meeting? Is your Year-3 plan well on track?

– Main accomplishments
  • Conversion of GFMC application code to use OpenMP
  • First large scale $^{12}$C calculations
  • Scaling to 8 racks on BG/P

– Grappling with scaling problems going from 8K to 32K processes

What are the aspects of your science that require high-performance computing? OR What problems in high performance computing are you working on in general?

– Problems in high-performance computing:
  • How to exploit HPC computers with 100,000 processors
  • How to simplify application programming in general
  • ADLB is a demonstration of what can be achieved with a semi-specialized library
The Questions (cont.)

- What are the major computational issues? Are there any questions you would like to bring to the attention of our CS/AM collaborators? OR Are there general capabilities of your computer science work that might be of interest to other physicists than the ones you are currently working with?
  - ADLB is a general-purpose library which we are developing / testing / debugging / tuning in the context of GFMC
  - But worth a look for any application in which the parallelism is task-based and there is little communication among the tasks.
  - ADLB Web site: http://www.cs.mtsu.edu/~rbutler/adlb

- What is the detailed roadmap of your project for the remaining part of Year-3 and Year-4? Could you sketch the work plan for Year 5?
  - Near-term: get to 16 racks, maybe 32, with good efficiency scaling
  - Better tools for understanding behavior and performance
  - Far-term: explore use of MPI RMA to further distribute work

- Are there any "showcase" (i.e., of Nature/Science caliber) physics and computational questions that you are hoping to answer in Years 3 and 4?
  - It’s up to Steve!
Conclusions

- ADLB is a research project working its way toward being useful general-purpose software.
- More users sought, especially those with more straightforward applications than GFMC!
- Its point is to explore whether extreme scalability in an application can be achieved without extreme complexity in application code.
- Much has been learned, understood, and achieved in the first few years.
- But we are not finished.
  - Which is good, in a way 😊.