Effective (Ab)se of HPC with Non-parallelized Software

Brent Eskridge, PhD
Department of Computer Science and Network Engineering
Southern Nazarene University

October 6, 2010
Any abuse of Sooner was purely unintentional.
Any abuse of Sooner was purely unintentional.

I promise.
Overview

- PhD in Computer Science from OU
- Interested in:
  - Autonomous agents
  - Multi-agent systems
  - Machine learning
  - Evolutionary computation
- Exclusively simulation
- First in research group to use Sooner
- HPC made my research possible
Scientific Problem
Motivations for Initial Research

▶ Develop controllers for autonomous agents
▶ Authentic agent problems
→ Complex tasks
▶ Authentic solutions
→ Combination of techniques to solve
Motivations for Initial Research

- Develop controllers for autonomous agents
- Authentic agent problems
  → Complex tasks
  → Authentic solutions
  → Combination of techniques to solve
Complex CINE Tasks

CINE
- Concurrent
- Interfering
- Non-Episodic

Details
Multiple tasks actively being addressed
Complex CINE Tasks

CINE
- Concurrent
- Interfering
- Non-Episodic

Details
Tasks have competing goals and share the same action space
Complex CINE Tasks

- Concurrent
- Interfering
- Non-Episodic

Details
Tasks do not terminate and are always active
Complex CINE Tasks

CINE

- Concurrent
- Interfering
- Non-Episodic

Details
Tasks in the intersection are the most difficult
Complex CINE Tasks: Examples

Active Tasks

- GOAL SEEK
- COLLISION AVOIDANCE
- RUNAWAY
- FLOCKING
  - ALIGNMENT
  - COHESION
  - SEPARATION
Complex CINE Tasks: Examples

Active Tasks

- **GOAL SEEK**
- **COLLISION AVOIDANCE**
- **RUNAWAY**
- **FLOCKING**
  - **ALIGNMENT**
  - **COHESION**
  - **SEPARATION**
Complex CINE Tasks: Examples

Active Tasks

- GoalSeek
- CollisionAvoidance
- RunAway
- Flocking
  - Alignment
  - Cohesion
  - Separation
Complex CINE Tasks: Examples

Active Tasks

- GOAL SEEK
- COLLISION AVOIDANCE
- RUNAWAY
- FLOCKING
  - ALIGNMENT
  - COHESION
  - SEPARATION
Developing controllers for these tasks is difficult
Need to make development of controllers practical
State and action abstraction can help, but
What are the benefits/costs of abstraction?
Experiments

- Developed controllers using different levels of abstraction
- Controllers were learned using:
  - Reinforcement learning (RL)
  - Evolutionary computation (EC)
- A total of 72 different experiments
- Each experiment required 40 runs
Recent Research

- Parameter choice in EC is a **black art**
- Are these parameters good?
- Triple Parameter Hypothesis tries to predict, but
- Does it work for a variety of problems?
- A total of **23** experiments
- Each experiment required **4,400** runs
Using HPC to Accomplish the Science
Software Limitations

- Programming Java for 10 years
- ECJ project in Java
  - Multi-threaded
  - Not really useful for Sooner’s architecture
- Custom simulator in Java
- Sooner has an old version of Java installed
- Java and MPI didn’t mix
Options

1. Spend time parallelizing existing project
2. Rewrite in C++ and use MPI
3. Abandon hope
Options

1. Spend time parallelizing existing project
2. Rewrite in C++ and use MPI
3. Abandon hope
Options

1. Spend time parallelizing existing project
2. Rewrite in C++ and use MPI
3. Abandon hope

Parallelizing the software is only one option...
The Ace Up My Sleeve

- Remember the total number of runs?
  - **Simulation**: $72 \times 40 = 2,880$
  - **Parameters**: $23 \times 4,400 = 101,200$

- Why not parallelize the runs?
- More bookkeeping, but
- Won’t change working code
Scripting to the Rescue

- Need to deal with:
  - Submitting jobs
  - Identifying failed jobs
  - Organizing results
  - Analyzing results

- Scripts can do all these things
- A full program is too much
- Used Perl and Bash scripts
  - Bash for scripting command line
  - Perl for parsing and analysis
So, How Did I Abuse Sooner?

- It wasn’t my fault
- It was their scheduler
- Other jobs required $N$ nodes at once
- Mine took single nodes when available
- Kept recycling jobs on same nodes
- Other jobs were starved
Questions?