Automatic Selection of Loop Scheduling Algorithms Using Reinforcement Learning

Sumithra Dhandayuthapani[1,2], Ioana Banicescu[1,2], Ricolindo L. Cariño[2], Eric Hansen[1], Jaderick P. Pabico[1,2], Mahbubur Rashid[1,2]

[1]Department of Computer Science, and
[2]Engineering Research Center
Mississippi State University

Partial support from the NSF Grants: 9984465, 0085969, 0081303, 0132618, and 0082979.

Scheduling and Load Balancing @ MSU

Motto: *Dynamic scheduling and load balancing algorithm development for performance optimization in scientific computing*

Activities

- Derive novel loop scheduling techniques
  - Adaptive weighted factoring (2000, ’01, ’02)
  - Adaptive factoring (2000)
- Develop load balancing tools and libraries
  - For applications using: Threads; MPI; DMCS/MOL
  - Addn’l functionality of systems: Loci; Hector
- Improve the performance of applications
  - N-body simulations; CFD simulations; Quantum physics; Astrophysics; Computational mathematics, statistics

Motivation: Time-stepping applications with parallel loops

Sequential form

```
Initializations
do t = 1, nsteps
    ...
    do i = 1, N
        (loop body)
    end do
    ...
end do
Finalizations
```

Parallel form

```
Initializations
do t = 1, nsteps
    ...
    call LoopSchedule (
        1, N,  loop_body_routine,
        myRank, foreman, method,
        ...)
    ...
end do
Finalizations
```

Property: The loop iterate execution times

(1) are non-uniform, and

(2) evolve with *t* .

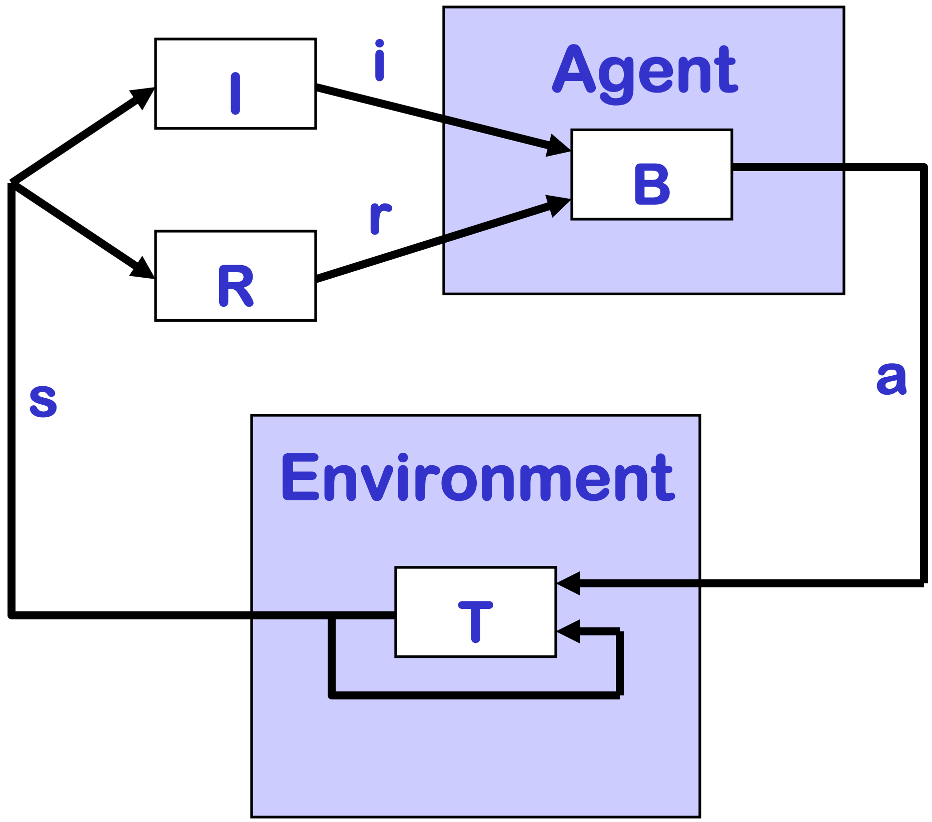
Problem: How to select the scheduling method?

Proposed solution: Machine Learning!

Machine Learning (ML)

- Supervised Learning (SL)
  - Teacher
  - Learner
  - Input-output pairs
  - Training (offline learning)
- Reinforcement Learning (RL)
  - Agent
  - Environment
  - Action, state, reward
  - Learning concurrent with problem solving
  - Survey: http://www-2.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/rl-survey.html

Reinforcement learning system



I – set of inputs (i)

R – set of rewards (r)

B – policy

a – action

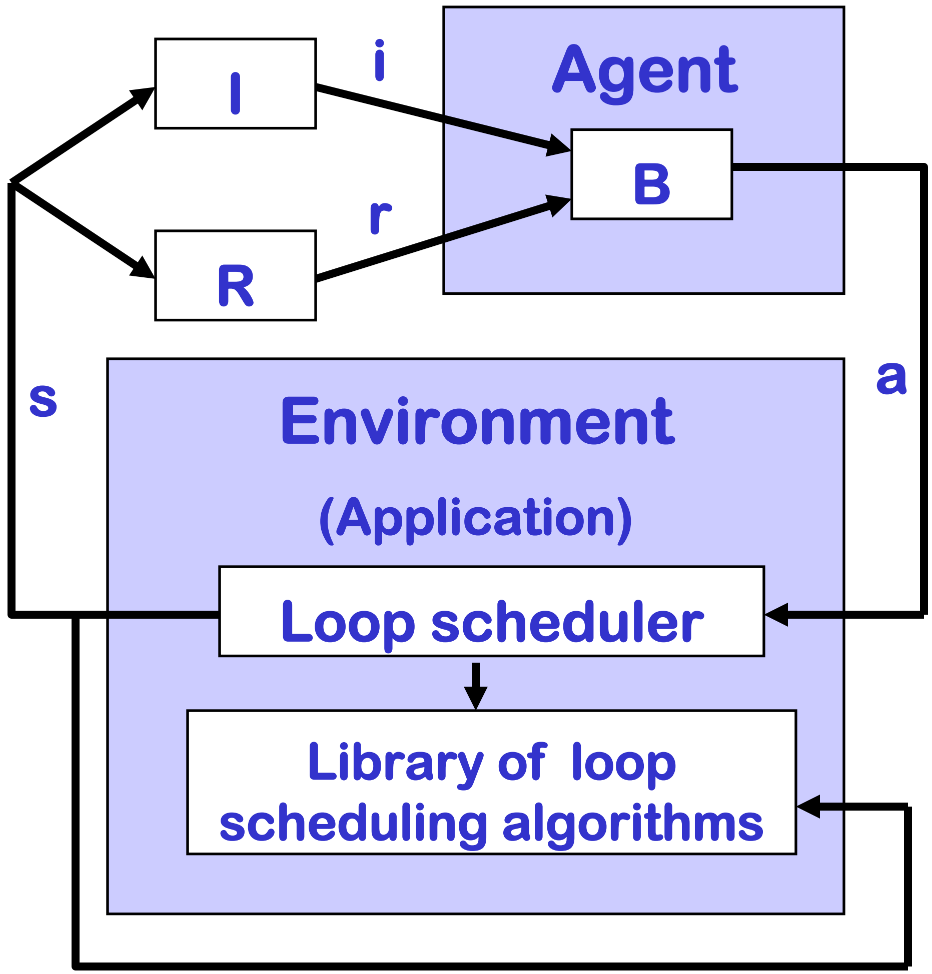
T – transition

s - state

Reinforcement Learning (RL)

- Model-based approach
  - Model *M*, utility function U_M from *M*
  - Examples: Dyna, prioritized sweeping, Queue-Dyna, Real-Time Dynamic Programming
- Model-free approach
  - Action-value function *Q*
  - Example: Temporal Difference (Monte Carlo + Dynamic Programming)
    - SARSA algorithm
    - Q Learning algorithm

RL system for automatic selection of loop scheduling methods



I – set of inputs (methods, time step, loop ids)

R – set of rewards (loop time)

B – policy (SARSA, Q)

a – action (use *method*)

s – state (application is using *method*)

Embedding a RL system in a time-stepping application

Serial form

```
Initializations
do t=1,nsteps
   ...
   do i = 1, N
      (loop body)
   end do
   ...
end do
Finalizations
```

Parallel form

```
Initializations
do t=1, nsteps
   ...
   call LoopSchedule(
      1, N, loop_body_rtn,
      myRank, foreman,
      method, ...)
   ...
end do
Finalizations
```

With RL system

```
Initializations
call RL_Init()
do t = 1, nsteps
   ...
   time_start = time()
   call RL_Action (method)
   call LoopSchedule (
      1, N, loop_body_rtn,
      myRank, foreman,
      method, ...)
   reward = time()-time_start
   call RL_Reward (t,
      method, reward)
   ...
end do
Finalizations
```

Test application: Simulation of wave packet dynamics using the QTM

- Bohm, D. 1952. “A Suggested Interpretation of the Quantum Theory in Terms of Hidden Variable,” *Phys Rev 85*, No. 2, 166-193.
- Lopreore, C.L., R.W. Wyatt. 1999. “Quantum Wavepacket Dynamics with Trajectories,” *Phys Rev Letters 82*, No. 26, 5190-5193.
- Brook, R.G, P.E. Oppenheimer, C.A. Weatherford, I. Banicescu, J. Zhu. 2001. “Solving the Hydrodynamic Formulation of Quantum Mechanics: A Parallel MLS Method,” *Int. J. of Quantum Chemistry 85,* Nos. 4-5, 263-271.
- Carino, R.L., I. Banicescu, R.K. Vadapalli, C.A. Weatherford, J. Zhu. 2004. “Message-Passing Parallel Adaptive Quantum Trajectory Method,” *High performance Scientific and Engineering Computing: Hardware/Software Support, L. T. Yang and Y. Pan (Editors)*. Kluwer Academic Publishers, 127-139.

Application summary

- **The time dependent Schrödinger's equation (TDSE)**

$$i\hbar\, \partial/\partial t\, \Psi = \mathrm{H}\, \Psi, \quad \mathrm{H} \equiv -(\hbar/2m)\nabla^2 + \mathrm{V}$$

  - quantum-mechanical dynamics of a particle of mass m moving in a potential V
  - $\Psi(r,t)$ is the complex wave function
- **The quantum trajectory method (QTM)**
  - $\Psi(r,t) = \mathrm{R}(r,t)\exp(i\,\mathrm{S}(r,t)/\hbar)$ (polar form; real-valued amplitude $\mathrm{R}(r,t)$, phase $\mathrm{S}(r,t)$ functions)
  - Plug $\Psi(r,t)$ into the TDSE, separate real and imaginary parts

$$-(\partial/\partial t)\rho(r,t) = \nabla \,.\, [\rho(r,t)(1/m)\nabla \mathrm{S}(r,t)]$$

$$-(\partial/\partial t)\mathrm{S}(r,t) = (1/2m)[\nabla \mathrm{S}(r,t)]^2 + \mathrm{V}(r,t) + \mathrm{Q}(\rho;\, r,t)$$

  - Probability density: $\rho(r,t) = \mathrm{R}^2(r,t)$
  - Velocity: $\mathrm{v}(r,t) = (1/m)\nabla \mathrm{S}(r,t)$
  - Flux: $\mathrm{j}(r,t) = \rho(r,t)\,\mathrm{v}(r,t)$
  - Quantum potential: $\mathrm{Q}(\rho;\, r,t) = -(1/2m)(\nabla^2 \log \rho^{1/2} + |\nabla \log \rho^{1/2}|^{\,2})$

QTM algorithm

Initialize wave packet x(1:N), v(1:N), ρ(1:N)

do *t = 1, nsteps*

do *i* = 1..N

call MWLS (*i*, x(1:N), ρ(1:N)*, p, b,*…); compute Q(*i*)

do *i* = 1..N

call MWLS (*i*, x(1:N), Q(1:N), *p, b,*…); compute f_q(*i*)

do *i* = 1..N

call MWLS (*i*, x(1:N), v(1:N), *p, b,*…); compute dv(*i*)

do *i* = 1..N

Compute V(i), f_c(i)

do *i* = 1..N

Update ρ(i), x(i), v(i)

Output wave packet

Embedding a RL system in a time-stepping application

Serial form

```
Initializations
do t=1,nsteps
   ...
   do i = 1, N
      (loop body)
   end do
   ...
end do
Finalizations
```

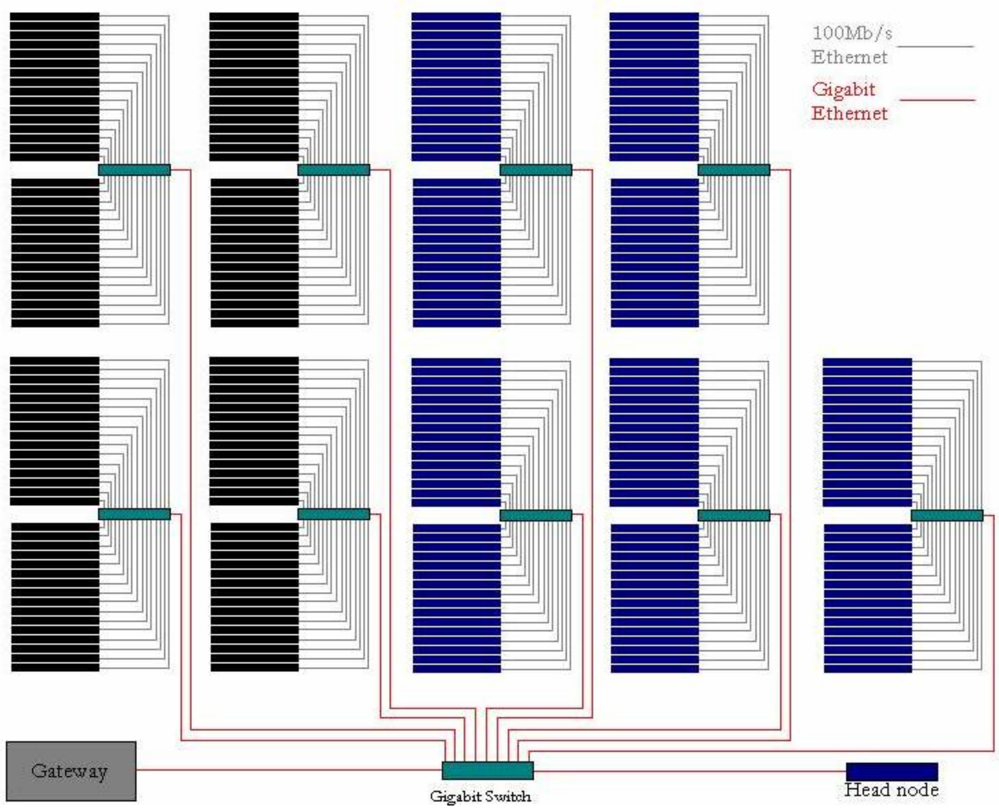
Parallel form

```
Initializations
do t=1, nsteps
   ...
   call LoopSchedule(
      1, N, loop_body_rtn,
      myRank, foreman,
      method, ...)
   ...
end do
Finalizations
```

With RL system

```
Initializations
call RL_Init()
do t = 1, nsteps
   ...
   time_start = time()
   call RL_Action (method)
   call LoopSchedule (
      1, N, loop_body_rtn,
      myRank, foreman,
      method, ...)
   reward = time()-time_start
   call RL_Reward (t,
      method, reward)
   ...
end do
Finalizations
```

Computational platform



- EMPIRE cluster
  - 1038 Pentium III (1.0 or 1.266 GHz)
  - Linux RedHat; PBS
  - 126th of Top 500 in 2002
- QTM in Fortran90, MPICH
- RL agent in C

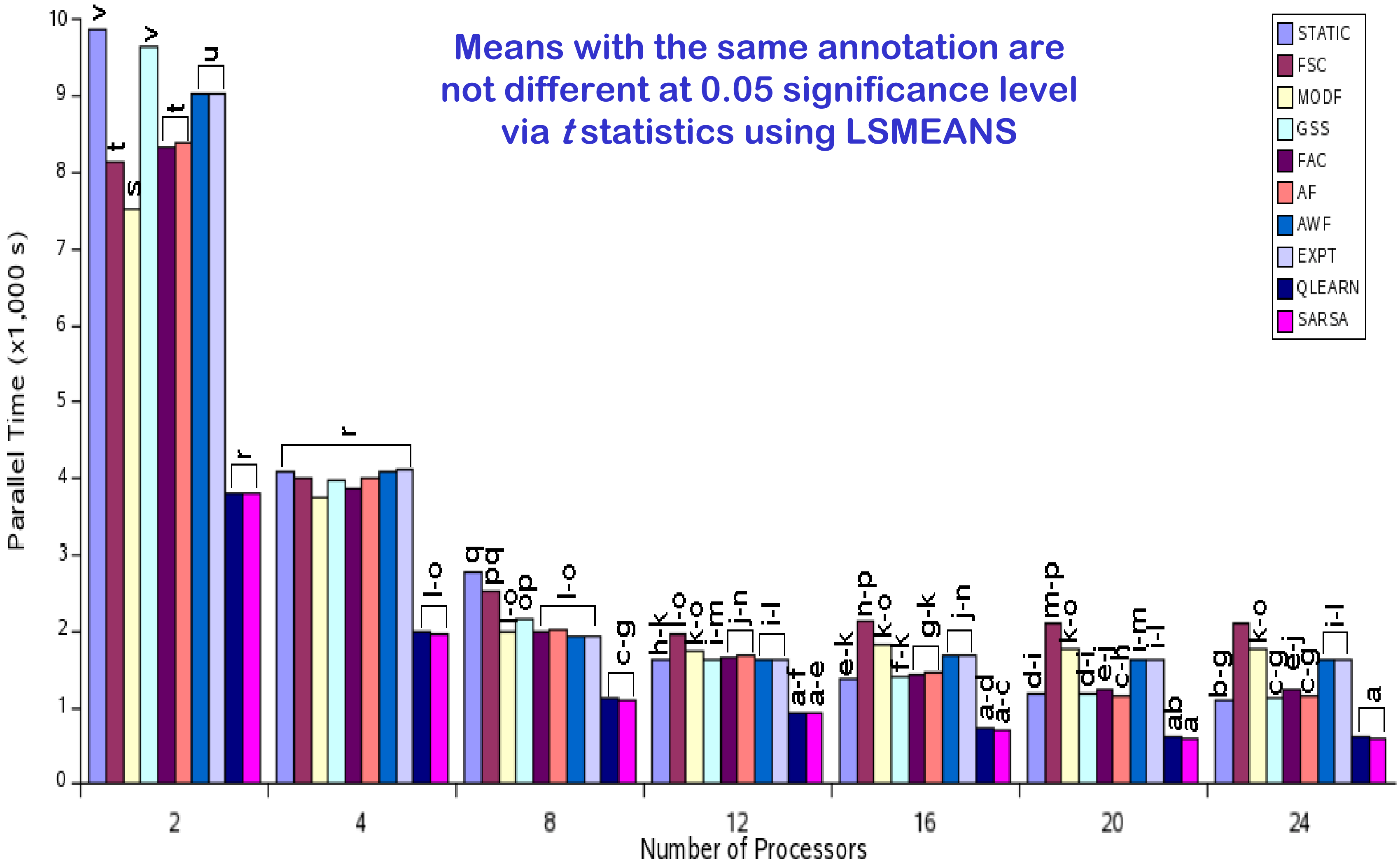
Experimental Setup

- Simulations
  - Free particle; harmonic oscillator
  - 501, 1001, 1501 pseudo-particles
  - 10,000 time steps
- No. of processors: 2, 4, 8, 12, 16, 20, 24
- Loop scheduling methods
  - Equal size chunks (STATIC, SELF, FSC)
  - Decreasing size chunks (GSS, FAC)
  - Adaptive size chunks (AWF, AF)
  - Experimental methods (MODF, EXPT)
  - RL agent (SARSA, Q)

Experimental Setup (cont)

- Hypothesis
  - The simulation performs better with RL than with a fixed scheduling method
- Design
  - Two-factor factorial experiment (factors: methods, no. of processors)
  - Five (5) replicates
  - Average parallel execution time T_P
  - Comparison via t statistic at 0.05 significance level, using Least Squares Means

Mean T_P of free particle simulation , 10000 time steps, 501 pseudo particles



Concluding remarks

- RL agent & loop scheduling library: suitable for time stepping applications with parallel loops
- RL agent consistently outperforms fixed methods in wave packet simulations
- Ongoing studies
  - No. of times a method was chosen by the RL agent?
  - Parametric study of SARSA, Q Learning
  - Other learning policies?
- RL in other time-stepping applications that require algorithm selection?

Contact:

Ioana Banicescu (ioana@cse.msstate.edu)