Application-Centric Metacomputing

Francine Berman

Department of Computer Science

University of California, San Diego

The Evolving Frontier

High-Performance Applications	Resource Reqts./ Computing Envt.
Grand Challenge Applications (1991)	GigaFLOPS machines Mega-Terabytes data + computers + viz
Supercomputer Center Apps (1997)	TeraFLOPS machines Terabytes of data + smart instruments + computers + data + viz
PetaFLOPS Apps (20??)	PetaFLOPS machines Tera-Petabytes dist. data + fast I/O + viz

Metacomputing

For resource-intensive applications, single machine may not be able to provide

- enough aggregate memory
- enough CPU
- enough storage
- support for appropriate computational paradigms

Metacomputing is the coordination of distributed resources to support parallel applications.

Metacomputing required to support resource-intensive, high-performance applications.

Using Metacomputing to Improve Performance

3D React

- Developed by Wu and Kupperman at CalTech

- Want to compare first principles chemical reaction model of $D + H_2 \dashrightarrow DH + H$ with simulation

- Simulation includes Berry's geometric phase to correct for local rotation

- Wu + Kupperman's code with Berry's phase correlates well with experimental data

3D REACT Simulation

Phase 1

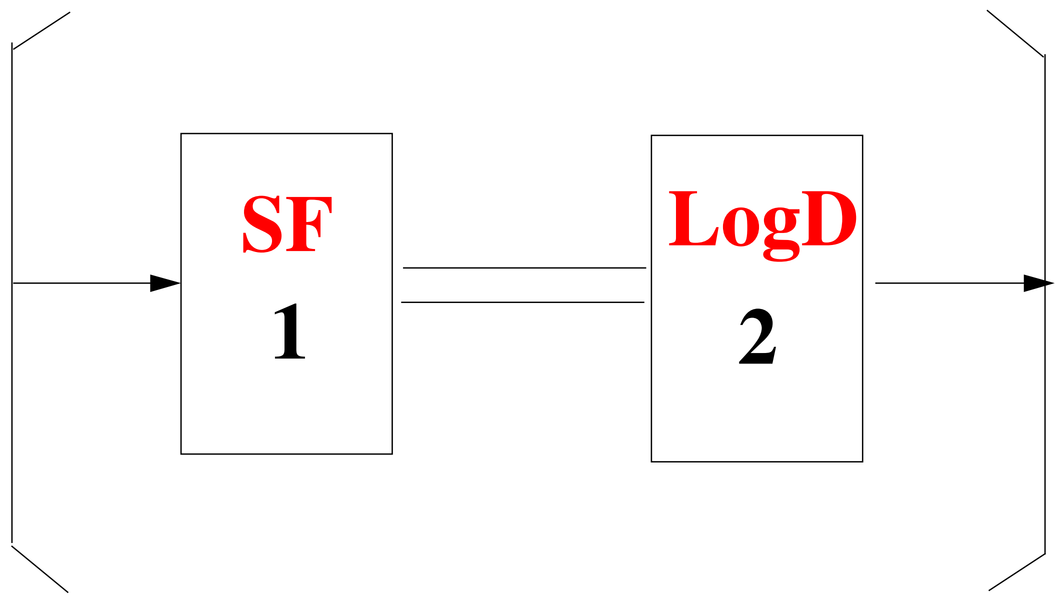
Generation of a set of surface functions for the expansion of the time-independent Schroedinger equation.

Phase 2

Solution of a coupled ordinary differential equation by log-derivative propagation methods.

Phases 1 and 2 **iterate** until integral and differential cross sections are obtained at a given total energy.

3D REACT Zoom Representation



3D REACT Metacomputing Implementation

On a heterogeneous network of two machines:

- Phase 1 vectorizes but has many sequential operations; efficient performance obtained on the C90
- Phase 2 involves matrix inversion and runs most efficiently on MPP
- Phase 1 and 2 can be executed concurrently; communication and computation can be overlapped
- Phases 1 and 2 typically executed for 360+ iterations

3D REACT Performance Results

Program run on **CalTech Delta** and **SDSC C90** interconnected by dedicated HiPPi link

C90 (1 CPU)	17.02 hours
Delta (64 nodes)	16.26 hours
Heterogeneous	**4.97 hours**

Speedup due to concurrent execution of the components and efficient matching of code to machines.

Metacomputing Applications

Resource-intensive applications developed as part of

- I-WAY at Supercomputing '95
- NSF Supercomputer Partnerships '97
- DOE ASCI program
- Next Generation Internet Initiative
- PetaFLOPS Initiative
- Business
  - financial modeling
  - inventory control
  - entertainment

The number of applications which require distributed resources are increasing ...

Enabling Metacomputing

Infrastructure must be developed to coordinate distributed resources

Many problems must be addressed:

- resource management
- scheduling
- security
- software and hardware heterogeneity
- separate administrative domains

Evolving infrastructure efforts:

- Globus (Argonne and ISI, US)
- Legion (U. of Virginia, US)
- MOL (U. of Paderborn, Germany)

Programming for Performance

Implementation of performance-efficient metacomputing application requires

- careful coordination of application resource requirements with system characteristics
- development of realistic models of both application and system performance
- accurate predictions of system load and availability *at execution time*

Scheduling is fundamental to the achievement of application performance.

Application Scheduling

Scheduling parallel applications involves

- *Mapping* application tasks to resources
- *Co-locating* data and computation
- *Scheduling tasks* on an individual resource
- *Scheduling communication* between resources

In **metacomputing environments**, this task is complicated by

- the **heterogeneity** of HW and SW
- dynamically varying load and availability of resources due to **contention** from other users

Scheduling for Application Performance

Why not use an MPP system scheduler?

- System scheduler must be in control of all applications
- System scheduler must be in control of all resources
- System scheduler optimizes for a fixed performance measure

Fundamental model of parallel system scheduler inadequate for metacomputing platforms.

How do users achieve performance on the metacomputer?

Currently users focus on

- dedicated time
- fixed resources
- static information

With evolving infrastructure

- local and wide-area networked systems will be used increasingly for **parallel** applications
- **interoperability** will be important
- **dynamic information** will be critical to performance
- focus will be on **application-level scheduling**

Application-Level Scheduling Paradigm:

Everything in the system is evaluated in terms of its impact on the application

Consequences of the Application-Level Scheduling paradigm:

- Both application-specific and system-specific information are required for good schedules.
- Performance depends upon the application's own performance criteria.
- The distance between resources is dependent upon how the application uses them.
- Predictions are accurate only within a particular time frame.
- A schedule is only as good as its underlying prediction.

Application-level Scheduling

Although metacomputing applications are different, the **same principles** are applied to achieve performance

The *Application-level Scheduling Paradigm* provides a **framework** for application scheduling on the metacomputer:

The underlying principles of application-level scheduling form the basis of the **AppLeS** approach to metacomputing application scheduling.

AppLeS

Joint project with Rich Wolski.

AppLeS = *Application-Level Scheduler*

Agent-based application scheduler based on the application-level scheduling paradigm

Application + AppLeS = Application'

Goal: **Enhance the user's scheduling process by**

- gathering and using more information
- evaluating more alternatives
- making better predictions
- using past performance history
- generating solutions *at machine speeds*

AppLeS Architecture

AppLeS Subsystems

The Coordinator manages 4 subsystems:

- Resource selector -- chooses and filters viable resource combinations
- Planner -- generates a schedule for a viable resource combination
- Performance estimator -- generates a performance prediction for candidate schedules
- Actuator -- interacts with the underlying resource management system to implement the schedule

AppLeS Data Sources

AppLeS uses a variety of **information sources**:

- Network Weather Service -- for information on the current and forecasted state of system resources
- Application-specific information -- provided by the user to characterize the structure of the computation
- User specifications -- to delineate the constraints, preferences and performance criteria of the user
- Structural performance models -- to allow the user and the scheduler to determine a desirable level of performance.

The Network Weather Service

Developed by Rich Wolski.

The **Network Weather Service** (NWS)

- senses resource performance throughout the system
- forecasts future performance of each resource at schedule time
- reports the forecast information to each AppLeS

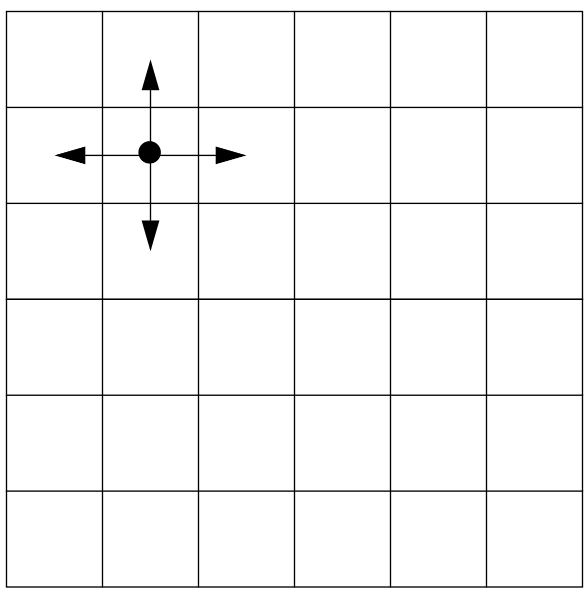
The NWS must **trade off**

- *intrusiveness* of sensors *vs. precision* of measurements
- *accuracy vs. complexity* of forecasts

Network Weather Service Architecture

A Jacobi 2D AppLeS

- Iterative solution to the LaPlace equation
- 2D regular grid, stencil communication for updates
- Code written by Wolski in KeLP/MPI
- performance criteria: minimal execution time
- **time-balancing** used to achieve performance
- NxN grid decomposed into **strips** for prototype



Francine Berman

PCL/SDSC Platform

Jacobi AppLeS Coordinator

Jacobi + AppLeS = Application

Blueprint:

1. Select candidate resource sets S_i

2. For each resource set S_i do

    Plan a schedule for S_i (strip decomposition)

    Estimate the cost of the schedule for S_i

3. Determine resource set and schedule with minimum predicted execution time.

4. Actuate the selected schedule.

Jacobi AppLeS Resource Selector

Goal: Narrow possibilities to a small collection of feasible resource sets.

Approach: Use filters, resource selection based on ***logical distance***.

Filters:

- access
- library availability
- memory

Logical Distance

What machines are considered "close" by the application?

locus = machine with the max value to user

list = sort of machines according to logical distance D

D(X,Y) = predicted execution time between X and Y for a unit partition

To determine resource list:

1. Let head = locus
   Let tail = locus
2. For i in resource set do
   - Find machine M such that D(tail,M) is minimum and M is not on the list
   - Add M to the tail of the list, tail = M

Using Structural Models to Predict Performance

Joint work with Jennifer Schopf

The structure of the model represents the operations where the application spends the most time.

- Can vary component models depending on their importance to application.
- Can instantiate the model to minimize error to a desirable level.

Jacobi AppLeS Performance Estimator

Performance model based on strip decomposition

$$T_i = \mathbf{Comp_i} + Comm_i$$

$$T_i = Area_i * Oper/pt + Comm_i$$

For the Jacobi AppLeS, we

- Solved system of linear equations for $Area_i$ based on the dynamic capacities of the available compute resources
- Set *Oper/pt* = $\boldsymbol{Oper/pt_{unloaded}*load}$ where *load* is the percentage of time P_i spends executing $Partition_i$.
- Determined $Comm_i$ as the time to send and receive data factored by dynamic BW (from the NWS).

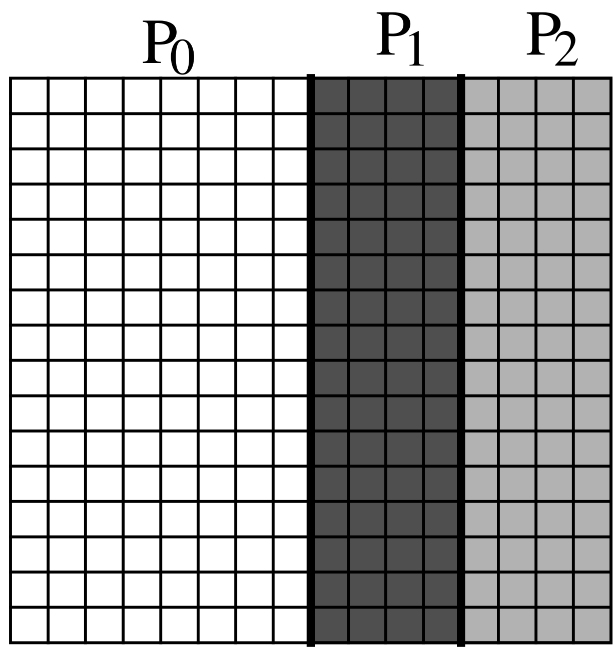
Jacobi AppLeS Planner

Planner solves the time-balancing equations to determine $\boldsymbol{Area_i}$

$$T_i = Area_i * Oper/pt + Comm_i$$

$$\Sigma\, Area_i = N*N$$

Result after rounding are strips $\{\boldsymbol{Area_i}\}$



Partitions rejected if $\boldsymbol{Area_i < 0}$ or $\boldsymbol{Area_i}$ does not fit in free memory.

Jacobi AppLeS Strip Decomposition

Jacobi AppLeS Actuator

Actuator implements schedule by passing mapping of data strips to processors to **KeLP**

KeLP (*Baden and Fink*) provides support for run-time data decomposition

KeLP and MPI serve as underlying resource management system

KeLP URL:

http://www-cse.ucsd.edu/groups/hpcl/scg/kelp.html

Experiments

We compared the performance of Jacobi 2D using

- Compile-time Block [HPF] partitioning
- Compile-time AppLeS partitioning
  [*no NWS, no resource selection*]
- Run-time AppLeS partitioning
  [*no resource selection*]
- Full run-time AppLeS partitioning

Method:

- Runs for different partitioning methods performed consecutively on production system
- Average execution time/iteration recorded.
- Used PCL/SDSC platform: Sparcs, RS6000s, Alpha Farm, SP-2

Results

All processors used for every run.

More Results

Resource selection used.

Gateway booted between PCL and SDSC at n=1800.

When Memory is Considered

Resource selection used.

Platform includes unloaded SP-2 processors; the best schedule uses these.

AppLeS and Metacomputing

Application-level scheduling *provides a paradigm for leveraging the resources of a metacomputing environment to achieve application performance.*

AppLeS prototype demonstrates that

- ✔ application-specific information
- ✔ dynamic information
- ✔ user's preferences and performance criteria
- ✔ prediction

are **all** required to achieve *application performance*.

Metacomputer Application Scheduling

How does AppLeS address the fundamental problems of metacomputing application scheduling?

Heterogeneity

- Load and availability cause all resources to exhibit individual performance characteristics
- **All** resources assumed to be heterogeneous

Contention

- Dynamic information provides an assessment of the fraction of resources available
- Available fraction can be regarded as contention-free and potentially **dedicated** to the application.

Current Status

AppLeS is a work in progress.

AppLeS Projects

- Development of AppLeS for a spectrum of applications
  - *Synthetic Aperture Radar Atlas (SARA)*
  - *Protein Docking application (DOT)*
  - *Shared Run Queue applications*
  - *Genetic Algorithm application*
  - *NetSolve AppLeS*
  - *Rescheduling AppLeS, etc.*
- Development of AppLeS on **Globus/GUSTO**, **Legion/DOCT** and **NPACI** testbeds

AppLeS Projects

Rescheduling

When is it profitable to redistribute the application?

$$T_{old} > T_{new} + T_{resched} + threshold$$

When should rescheduling be done?

- system cues
- application cues/unit of progress
- user cues
- scheduler cues

How does the "*quality of information*" of the predictions affect the rescheduling decision?

AppLeS Projects

Bushel of AppLeS Problem:

What happens when multiple AppLeS want to use the same resources?

Need to consider

- resource reservation/quality of service
- market economy approach
- system-imposed "social structure"

Project Information

AppLes Corps:

Fran Berman (berman@cs.ucsd.edu)

Marcio Faerman

Chris Peterson

Gary Shao

Jenny Schopf

Neil Spring

Glenn Reinman

Rich Wolski (rich@cs.ucsd.edu)

AppLeS Home Page:

http://www-cse.ucsd.edu/groups/hpcl/apples/apples.html