What I Mean By ...

- “High-performance Java” (HP/J)
  - Large-scale, multi-machine
  - Parallel applications:
    - Multi-threaded (e.g., shared memory HPC machine) or RMI-based or message-passing parallelism
    - CPUs/OS typically homogeneous
  - Distributed applications:
    - Distributed JVM or cooperating JVMs
    - Typically heterogeneous
The Idea of HP/J Is Gaining in Popularity

- Portability
- Higher abstraction than previous parallel computing
- Power as important / more important than performance

What I Mean By ...

- "HP/J tools"
  - Software or middleware outside the application
  - Does not include programming classes, libraries, templates, etc.

- A number of IDEs are available, e.g.
  - Java WorkShop (Sun)
  - Visual Age for Java (IBM)
  - Jbuilder (Borland)
  - Visual Café (Symantec)
- They do more than IDEs for other languages
  - Support for object classes, event model, concurrent threads, beans
- ... But they do not really address HP/J
  - No provision for multiple CPUs, thread migration, etc.
  - Assume a single JVM

NACSE - Oregon State
User Requirements for HP/J Tool Support

Requirement 1: Launching and controlling threads that execute on different JVMs

Tools that Create/Launch Multi-JVM Applications

- Approach A: explicit assignment of processors
  - JavaPorts launches application, manages “port operations”
  - Requires script to assign “tasks” and allocate processors

begin configuration
begin definitions
  define machine M1="corea.cdsp.neu.edu" master
  define machine M2="walker.cdsp.neu.edu"
  define task T1="Manager" numofports=1
  define task T2="Worker1" numofports=1
  define task T3="Worker2" numofports=1
  define task T4="Worker3" numofports=1
end definitions
begin allocations
  allocate T1 M1
  allocate T2 M2
  allocate T3 M2
  allocate T4 M3
end allocations
begin connections
  connect M1.T1.P[0]  M2.T2.P[0]
end connections
end configuration
Tools that Create/Launch Multi-JVM Applications

- Approach B: automatic assignment of processors
  - Users select application from list, then assign input files
  - **DOGMA** selects processors and launches application

User Requirements for HP/J Tool Support

Requirement 2: Examining the progress of threads that execute on different JVMs
Examining the Progress of HP/J Applications

• Approach A: text-based (command-driven) debugging
  - **SDB** (UC Berkeley) for Titanium dialect of Java
  - **DCDB** (National HPC Center, Hefei, China)
• Provides simple functionality (breakpoints, single-stepping, info on threads, info on JVMs)
  ... but interfaces are reminiscent of 1970s debuggers

```
(sdb) break worker.ti:30
Breakpoint 1 at 0x33dd0: file worker.ti, line 30.
(sdb) run
Starting program: Srvc2
Breakpoint 1, Main.main (args=0x197f80) at worker.ti:30
30     aPoint = [2];
(sdb) thread 12
[Switching to Thread 5 (LWP 4)]
#0  Main.main (args=01xcaf00) at Worker.ti:30
30  aPoint = [2];
(sdb) print ti_array[aPoint]
$8 = (MyClass) 0x1a1648 (java.lang.Object.PrivateInfo = 0,
MyClass.x = 1, MyClass.y = 2
(sdb)
```

Examining the Progress of HP/J Applications

• Approach B: paned text-based debugging
  - **Metamata Debug** (Metamata Inc.)
  - Provides multiple, tabbed windows for controlling and viewing each thread
  - Also adds code and object browsers
  ... but still limited to rudimentary commands
User Requirements for HP/J Tool Support

Requirement 3: Stabilizing errors occurring across JVMs so possible causes can be explored

Stabilizing the Progress of HP/J Applications

• Approach: support for deterministic replay
  - DejaVu (IBM Research)
• Use JVM extensions to capture sequence of execution
  - Makes it possible to replay execution later
  - When coupled with debugger, makes it possible to stabilize errors

“Maybe we should write that spot down.”
Tuning the Performance of HP/J Applications

- Several tool suites include support for performance analysis and tuning
  - Quantify and PureCoverage (Rational Software)
  - OptimizeIt!
  - Jprobe Profiler, Threadalyzer, Coverage, Memory Debugger (KL Group)
  - Numega TrueTime, Memory Profiler, JCheck, TrueCoverage (Compuware)
  - ... but they do not support multi-JVM applications

Current “Best-in-Class” for Single JVMs

- HPjMeter
  - Lets user choose how much information to see (top-10 only, etc.)
  - “Find” operations facilitate comparing method timings vs. counts, etc.
  - Provides “hints” about what to do next
  - Documentation is organized by user task
  - Can be downloaded free
  - [will see more later]
Requirement 4: Identifying the primary candidates for optimization activities

Tuning HP/J Applications

• Users don’t want to spend optimization efforts indiscriminately
Tuning HP/J Applications

- Profilers can help … but user still has to do a lot of work

User Requirements for HP/J Tool Support

Requirement 5: Identifying and eliminating performance bottlenecks occurring across JVMs
Finding Bottlenecks in HP/J Applications

• Dynamic call trees help identify bottlenecks
  ... but don't scale well to volume, method names in Java

  TAU trace + VAMPIR visualization

Finding Bottlenecks in HP/J Applications

• Collapsible trees scale better
  ... but make it hard to compare

  JavaViz (IBM Research)
User Requirements for HP/J Tool Support

Requirement 6: Identifying and eliminating potential synchronization problems across JVMs

Finding Bottlenecks in HP/J Applications

- Trace visualizations help identify synchronization bottlenecks
  ... but don’t scale to many threads/processes, long runs
User Requirements for HP/J Tool Support

Requirement 7: Understanding the implications of increasing parallelism

Determining Scalability of HP/J Applications

- Trace visualizations help show how performance changes as problems scale
  ... but don’t correlate with objects and classes
Today’s HP/J Tool Situation

- Today’s tools are
  - Clumsy to use
  - Hard to relate to objects and other abstractions
  - Unable to analyze even simple patterns
- They swamp the user with confusing detail
  ... and they don’t really help the user write better Java

Why Aren’t HP/J Tools Better?

- Today’s HP/J tools borrow heavily from earlier tools
  ... that were designed to meet other needs
Today's Tools “Borrow” from the Past

- Tools for creating/launching HP/J applications
  - Scripted and GUI-based launching: adapted from early parallel programming tools (GMAT, Schedule, PVM scripts)

- Tools for debugging HP/J applications
  - Text-based debugging: adapted from early serial debuggers (1970s)
  - Pane-based debugging: adapted from early parallel debuggers (1980s)
  - Deterministic replay: adapted from early parallel debuggers (1980s)

- Tools for tuning HP/J applications
  - Profilers: adapted with few changes from earliest parallel versions (early 1980s)
  - Trace-based tools: adapted with few changes from first parallel tools (late 1980s)
  - In general, ignore the nature and special requirements of Java
HP/J Tools Don’t “Borrow” Effectively Enough

- HP/J tools don’t provide the best functionality from IDEs

Recommendation 1:
Organize levels of detail into click-for-more hierarchy

HP/J Tools Don’t “Borrow” Effectively Enough

- HP/J tools report methods, but not object classes

Recommendation 2:
Facilitate exploring object patterns-of-reference
Recommendation 3:
Track activity/performance of objects per-instance

HP/J Tools Don’t “Borrow” Effectively Enough

- HP/J tools report method invocations, but not object instances

Recommendation 4:
Aggregate related threads/processes for display

HP/J Tools Don’t “Borrow” Effectively Enough

- HP/J traces are organized by process/thread #, not object groups
HP/J Tools Aren’t Exploring “New Approaches”

- Previous tools were designed for previous programming models
- **HP/J requires newer approaches:**
  - Metaphors for representing behavior of distributed objects
  - Ability to aggregate and abstract huge amounts of execution data
  - Visual portrayals that distinguish between threads and methods executing on different JVMs

Existing Tool Technologies Aren’t Enough

- **Previous Java IDEs**
  - Didn’t have to deal with multiple JVMs
  - Didn’t have to deal with true parallelism
  - Didn’t scale to the numbers needed for HP/J
  - Are still viewed as lacking in many respects

- **Previous parallel tools**
  - Forced programmers to think at low levels of detail
  - Didn’t support objects or distributed systems
  - Didn’t scale to the numbers needed for HP/J
  - Weren’t considered adequate by programmers, anyway

We shouldn’t be relying on previous tool approaches!
HP/J Tools Require “Newer Approaches”

- HP/J tools don’t address Java’s worst performance problem - synchronization bottlenecks

Recommendation 5:
Explicitly identify where access patterns cause bottlenecks

HP/J Tools Require “Newer Approaches”

- HP/J tools are organized in terms of low-level mechanisms, not user goals

Recommendation 6:
Phrase options in terms of problems being addressed
HP/J Tools Require “Newer Approaches”

- HP/J tools don’t help users find changes from one execution to another

What Should We Expect from HP/J Tools?

- A tool that is poorly suited to the task ... is as bad as no tool at all! (only suitable for “geniuses”)
We Need Tools That …

- Reflect largescale and distributed parallelism
- Move programmers to higher levels of abstraction
- Adapt to the particular nature of Java
  - Variability in per-instance performance of objects
  - Unpredictable lifespan of threads
  - Hidden impact of bottlenecks
- Don’t just re-work old mechanisms and metaphors

What Should We Expect from HP/J Tools?

- Too much reliance on the past is limiting our future!
Special Project: Performance Database

- **Goal:** Create a permanent archive for results from benchmark runs
  - Web-based access to tracefiles in standard format(s)
  - Benchmark source code (where not proprietary)
  - Metadata describe key aspects of program run
  - Query by setup conditions, event types, etc.
  - Download into spreadsheets or private databases
- **Status**
  - (Initially) Joint project of NACSE and Univ.di Pavia
  - Funded by U.S. Dept. of Defense HPC Modernization Program
  - Database now contains most of Pavia’s SPEC tests
- **Future**
  - Looking for other partners
  - Feedback: Should we add profiler output as well as tracefiles?

Produced with (involuntary) artistic assistance from:

Gary Larson, creator of “Far Side”
Newspaper cartoonists John O’Brien,
S. Harris, and Goff