Panel Session:
Will Software Save Moore’s Law?

Vivek Sarkar

(vsarkar@us.ibm.com)

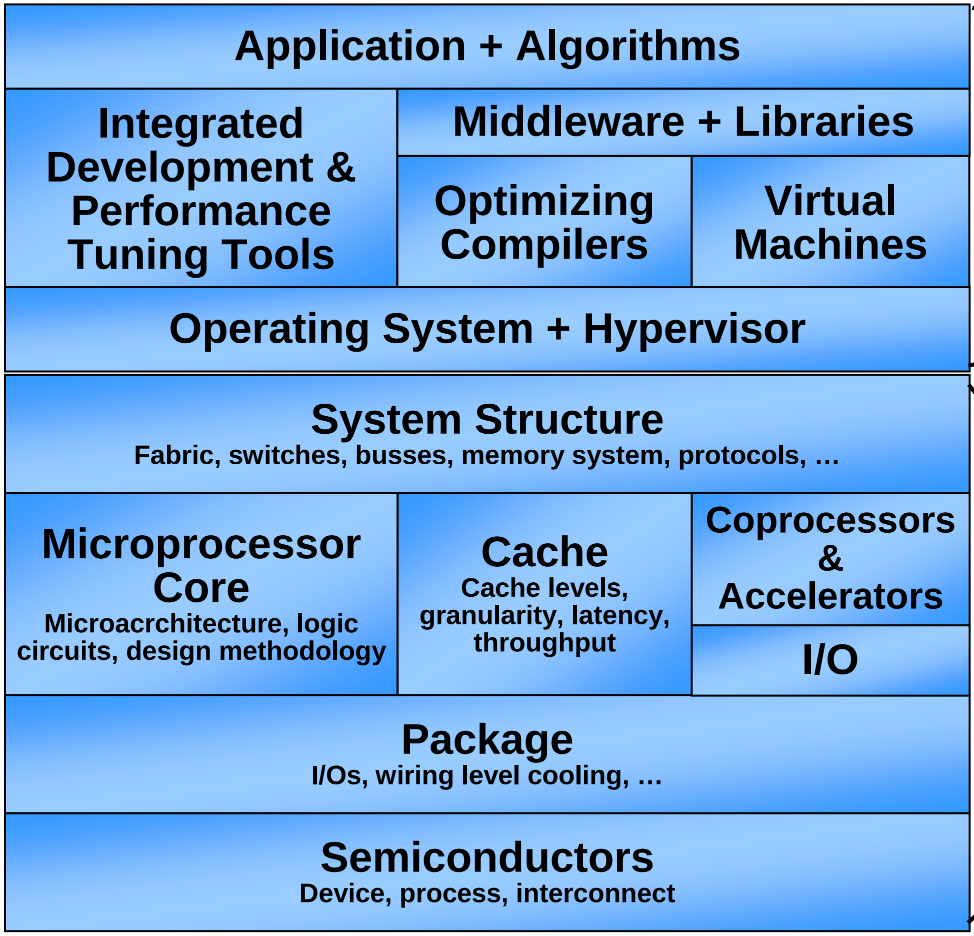
Senior Manager, Programming Technologies

IBM T.J. Watson Research Center

This work has been supported in part by the
Defense Advanced Research Projects Agency (DARPA)
under contract No. NBCH30390004.

System-level Performance of Embedded Applications

Goal: maintain 2x CAGR improvement every 1.5 years in system performance per unit power/area/cost/. . .



Application + Algorithms

Integrated Development & Performance Tuning Tools

Middleware + Libraries

Optimizing Compilers

Virtual Machines

Operating System + Hypervisor

System Structure
Fabric, switches, busses, memory system, protocols, …

Microprocessor Core
Microacrchitecture, logic circuits, design methodology

Cache
Cache levels, granularity, latency, throughput

Coprocessors & Accelerators

I/O

Package
I/Os, wiring level cooling, …

Semiconductors
Device, process, interconnect

Challenge: Software contribution must exceed 2x every 1.5 years!

Hardware contribution reducing to 2x every 3-5 years

Q1: Can software overcome Wirth's Law in time to save Moore's Law?

***<u>Answer:</u>* Yes! The only way to save Moore's Law is through parallelism, and the only way to exploit parallelism (beyond ILP) is through software.**

Scale-up Parallelism

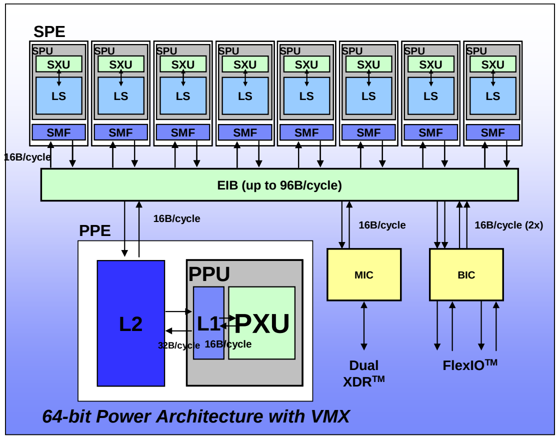
SMP

Multi-core

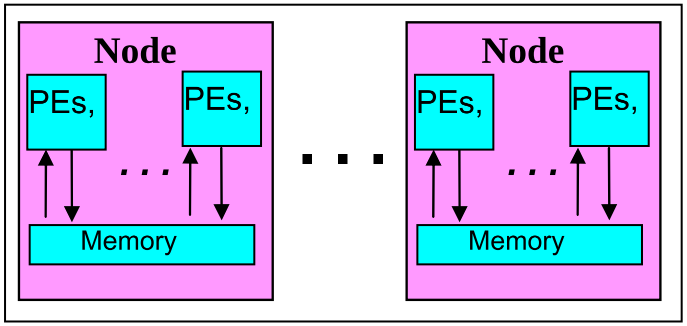
SMT's

SIMD

Heterogeneous Parallelism (Co-processors, accelerators)



Scale-out Parallelism



Parallelism in all the above cases can only be exploited through software: operating system, optimizing compilers, virtual machines, libraries, middleware, algorithms, applications, …

Q2: What benefits can we expect from the new high-productivity languages under development?

1) Safety -- eliminate entire classes of errors through static & dynamic safety checks

- Type errors, initialization errors, pointer errors, array indexing errors – no accesses to inconsistent data
- Branching errors – no branches to unpredictable code locations
- Higher-level semantic errors – deadlock, data races, writes to immutable data, violations of preconditions/postconditions

2) High level concurrency – integration of threads with a partitioned global address space and scalable synchronization operations

- Unified model for fine-grain multithreading, one-sided data transfers, remote atomic operations, active messages, …
- Relaxed memory model for synchronization operations

3) Optimized implementations on high-end HPC systems

- Early implementations of PGAS languages (UPC, CAF) point the way

4) Ecosystem: Integration with Environment, Libraries, and Tools

- Use of tools integrated into open & extensible development environments, such as Eclipse
- First-class support for components and libraries

Q3: Is it possible to use portable software with the latest hardware technologies? (Graphics processors, Cell, PCA, FPGAs, etc.)

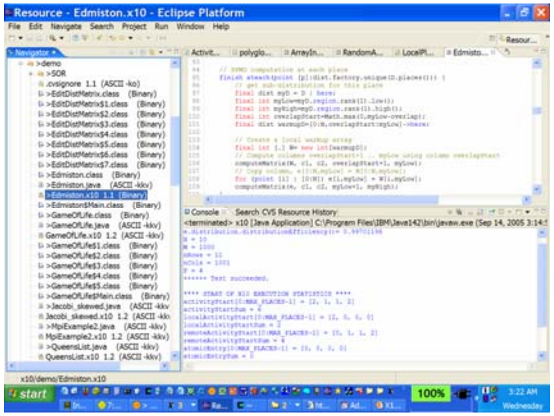
Answer: ***Yes!***
Virtual machines and integrated tools are the key to portability

HPC VM Enhancements --- IBM PERCS/X10 project

Real-time VM Enhancements --- IBM Metronome project

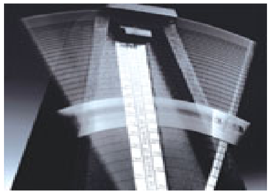
Commodity Virtual Machines

Eclipse as a foundation for common tools


Resource - Edmiston.x10 - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Resour...
Navigator
>demo
>SOR
.cvsignore 1.1 (ASCII -ko)
>EditDistMatrix.class (Binary)
>EditDistMatrix$1.class (Binary)
>EditDistMatrix$2.class (Binary)
>EditDistMatrix$3.class (Binary)
>EditDistMatrix$4.class (Binary)
>EditDistMatrix$5.class (Binary)
>EditDistMatrix$6.class (Binary)
>EditDistMatrix$7.class (Binary)
>Edmiston.class (Binary)
>Edmiston.java (ASCII -kkv)
>Edmiston.x10 1.1 (Binary)
>Edmiston$Main.class (Binary)
>GameOfLife.class (Binary)
>GameOfLife.java (ASCII -kkv)
GameOfLife.x10 1.2 (ASCII -kkv)
>GameOfLife$1.class (Binary)
>GameOfLife$2.class (Binary)
>GameOfLife$3.class (Binary)
>GameOfLife$4.class (Binary)
>GameOfLife$5.class (Binary)
>GameOfLife$Main.class (Binary)
>Jacobi_skewed.java (ASCII -kkv
Jacobi_skewed.x10 1.2 (ASCII -kk
>MpiExample2.java (ASCII -kkv)
MpiExample2.x10 1.2 (ASCII -kkv)
>QueensList.java (ASCII -kkv)
QueensList.x10 1.2 (ASCII -kkv)
Activit... polyglo... ArrayIn... RandomA... LocalPl... Edmisto...
Console Search CVS Resource History
x10/demo/Edmiston.x10
start
100%
3:22 AM
Wednesday

```
// SPMD computation at each place
finish ateach(point [p]:dist.factory.unique(D.places())) {
    // get sub-distribution for this place
    final dist myD = D | here;
    final int myLow=myD.region.rank(1).low();
    final int myHigh=myD.region.rank(1).high();
    final int overlapStart=Math.max(0,myLow-overlap);
    final dist warmupD=[0:N,overlapStart:myLow]->here;

    // Create a local warmup array
    final int [.] W= new int[warmupD];
    // Compute columns overlapStart+1 ... myLow using column overlapStart
    computeMatrix(W, c1, c2, overlapStart+1, myLow);
    // Copy column, e[0:N,myLow] = W[0:N,myLow];
    for (point [i] : [0:N]) e[i,myLow] = W[i,myLow];
    computeMatrix(e, c1, c2, myLow+1, myHigh);
```

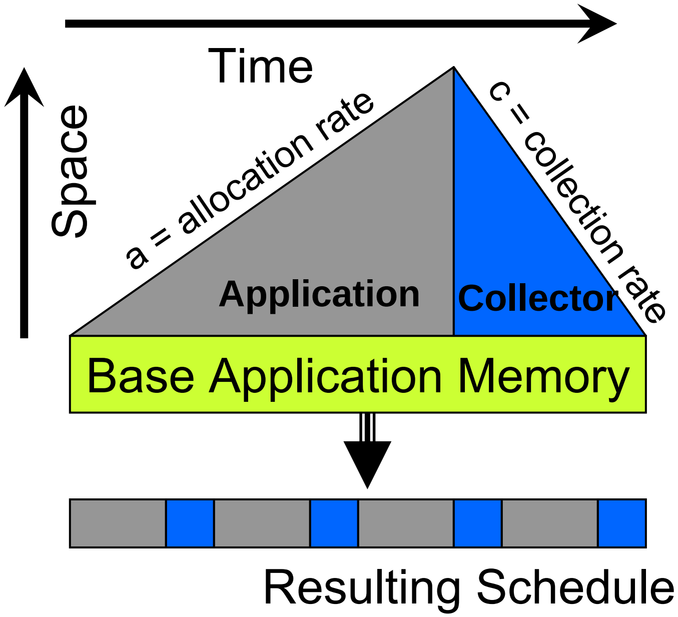
```
<terminated> x10 [Java Application] C:\Program Files\IBM\Java142\bin\javaw.exe (Sep 14, 2005 3:14:
e.distribution.distributionEfficiency() = 0.99701196
N = 10
M = 1000
nRows = 11
nCols = 1001
P = 4
****** Test succeeded.

**** START OF X10 EXECUTION STATISTICS ****
activityStart[0:MAX_PLACES-1] = [2, 1, 1, 2]
activityStartSum = 6
localActivityStart[0:MAX_PLACES-1] = [2, 0, 0, 0]
localActivityStartSum = 2
remoteActivityStart[0:MAX_PLACES-1] = [0, 1, 1, 2]
remoteActivityStartSum = 4
atomicEntry[0:MAX_PLACES-1] = [0, 2, 0, 0]
atomicEntrySum = 2
```



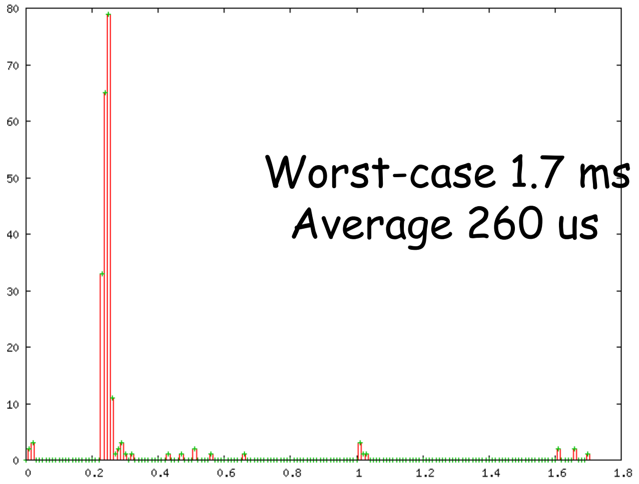
IBM Metronome project: Portable Real-time Garbage Collection

David Bacon, Perry Cheng, David Grove, V.T. Rajan, Martin Vechev

- **Garbage collection is fundamental to Java's value proposition**
  - **Safety, reliability, programmer productivity**
  - **But also causes the most non-determinism (100 ms – 10 s latencies)**
  - **RTSJ standard does not support use of garbage collection for real-time**
- **Metronome is our hard real-time garbage collector**
  - **Worst-case 2 ms latencies; high throughput and utilization**
  - **100x better than competitors' best garbage collection technology**

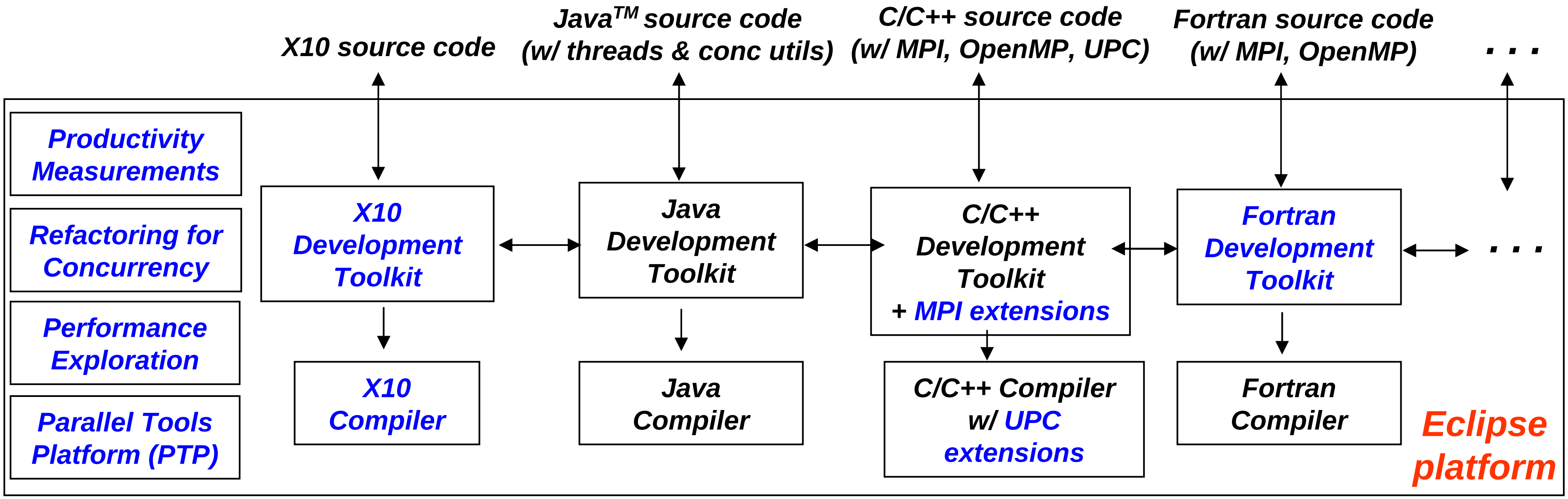


Garbage Collection Pause Times (Customer application)



Worst-case 1.7 ms
Average 260 us

PERCS Programming Model, Tools and Compilers
(PERCS = Productive Easy-to-use Reliable Computer Systems)


X10 source code

Java^{TM} source code (w/ threads & conc utils)

C/C++ source code (w/ MPI, OpenMP, UPC)

Fortran source code (w/ MPI, OpenMP)

. . .

Productivity Measurements

Refactoring for Concurrency

Performance Exploration

Parallel Tools Platform (PTP)

X10 Development Toolkit

Java Development Toolkit

C/C++ Development Toolkit + MPI extensions

Fortran Development Toolkit

. . .

X10 Compiler

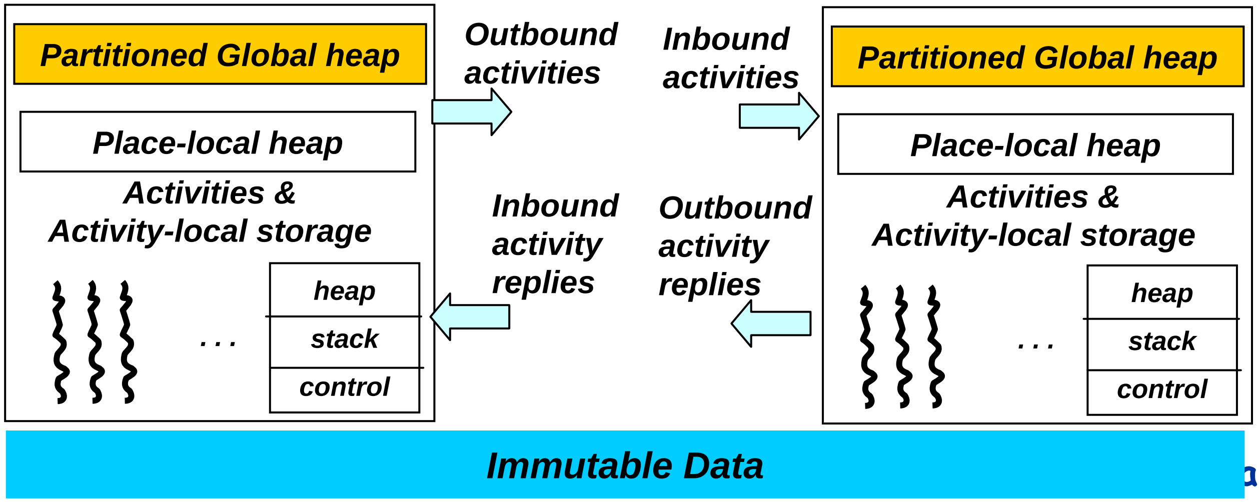
Java Compiler

C/C++ Compiler w/ UPC extensions

Fortran Compiler

Eclipse platform

Text in blue identifies exploratory PERCS contributions


Partitioned Global heap

Place-local heap

Activities & Activity-local storage

heap

stack

control

. . .

Outbound activities

Inbound activities

Inbound activity replies

Outbound activity replies

Partitioned Global heap

Place-local heap

Activities & Activity-local storage

heap

stack

control

. . .

Immutable Data

X10 Threaded Partitioned Global Address Space model (T-PGAS)