A Performance Estimator ---

PetaSIM

and its relationship to Application Emulators

Geoffrey Fox and Yuhong Wen

Northeast Parallel Architecture Center (NPAC)

Syracuse University

May 18, 1998

PetaSIM Motivation

- **PetaSIM was designed to allow “qualitative” performance estimates” where in particular the design of machine is particularly easy to change**
- **The project will build up a suite of applications which can be used in future activities such as “Petaflop” architectures studies**
- **Applications are to be derived “by hand” or by automatic generation from Maryland Application Emulators**
- **Special attention to support of hierarchical memory machines and data intensive applications**
- **Support parallelism and representation at different grain sizes**
- **Support simulation of “pure data-parallel” and composition of linked modules**

PetaSIM Interpolates Between

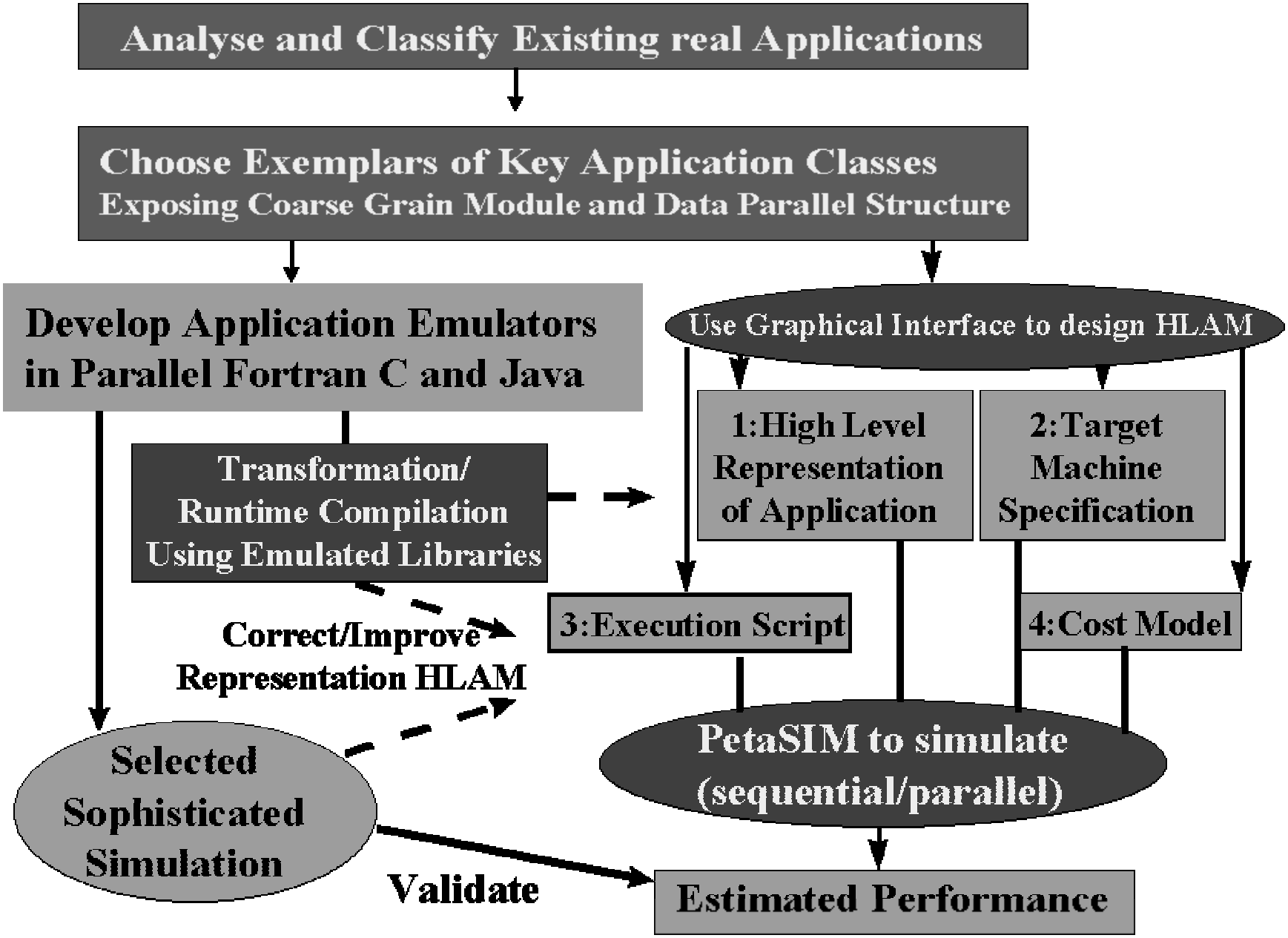
- **"Back of the envelope" where you get good intuition for why the performance is what it is**

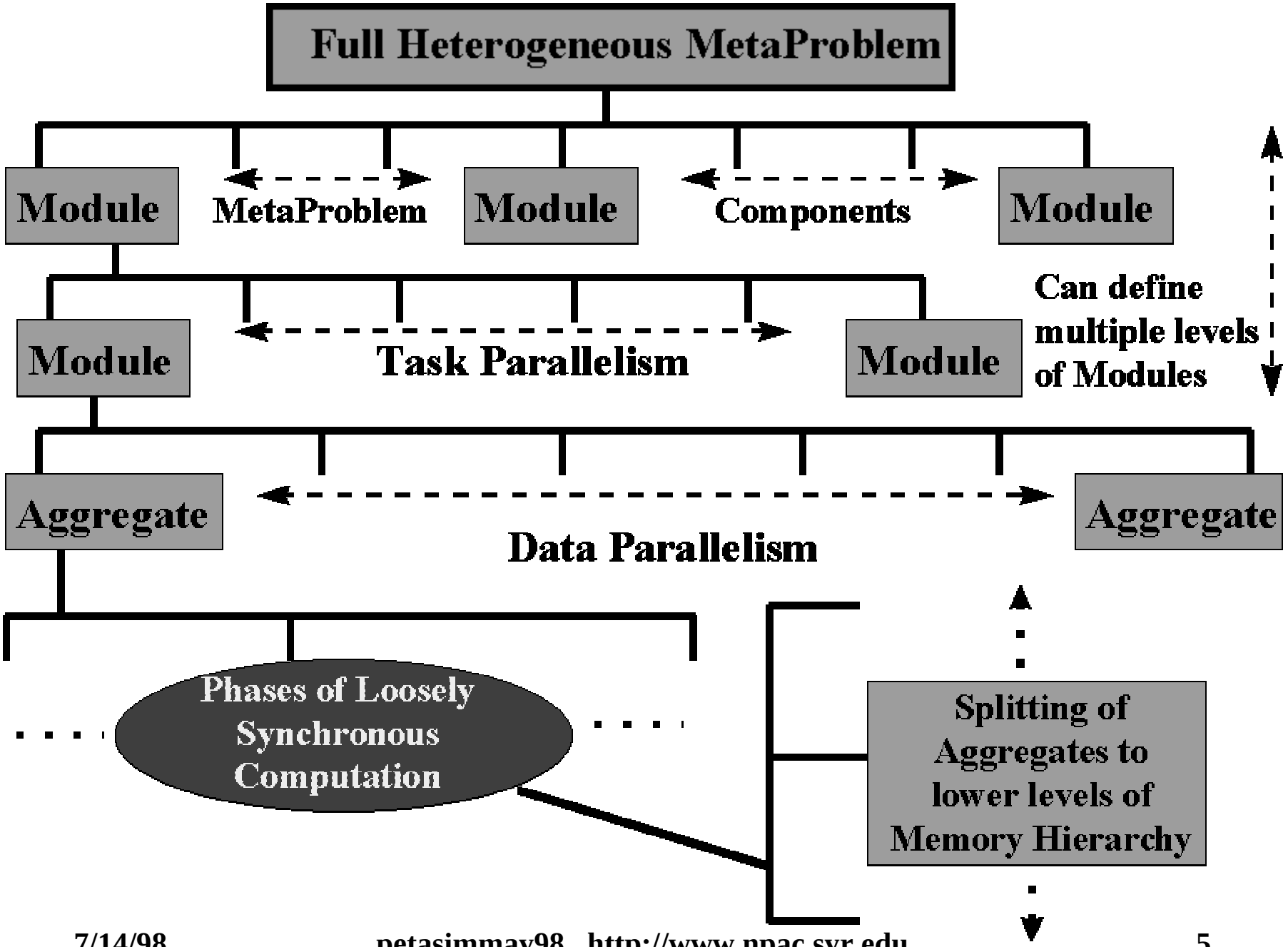
Communication Overhead in classical data parallel (edge over area) with NO memory hierarchy NO latency

$$\alpha \frac{t_{comm}}{t_{float}\,(\text{grain size})^{1/dim}}$$

where dim is geometric dimension

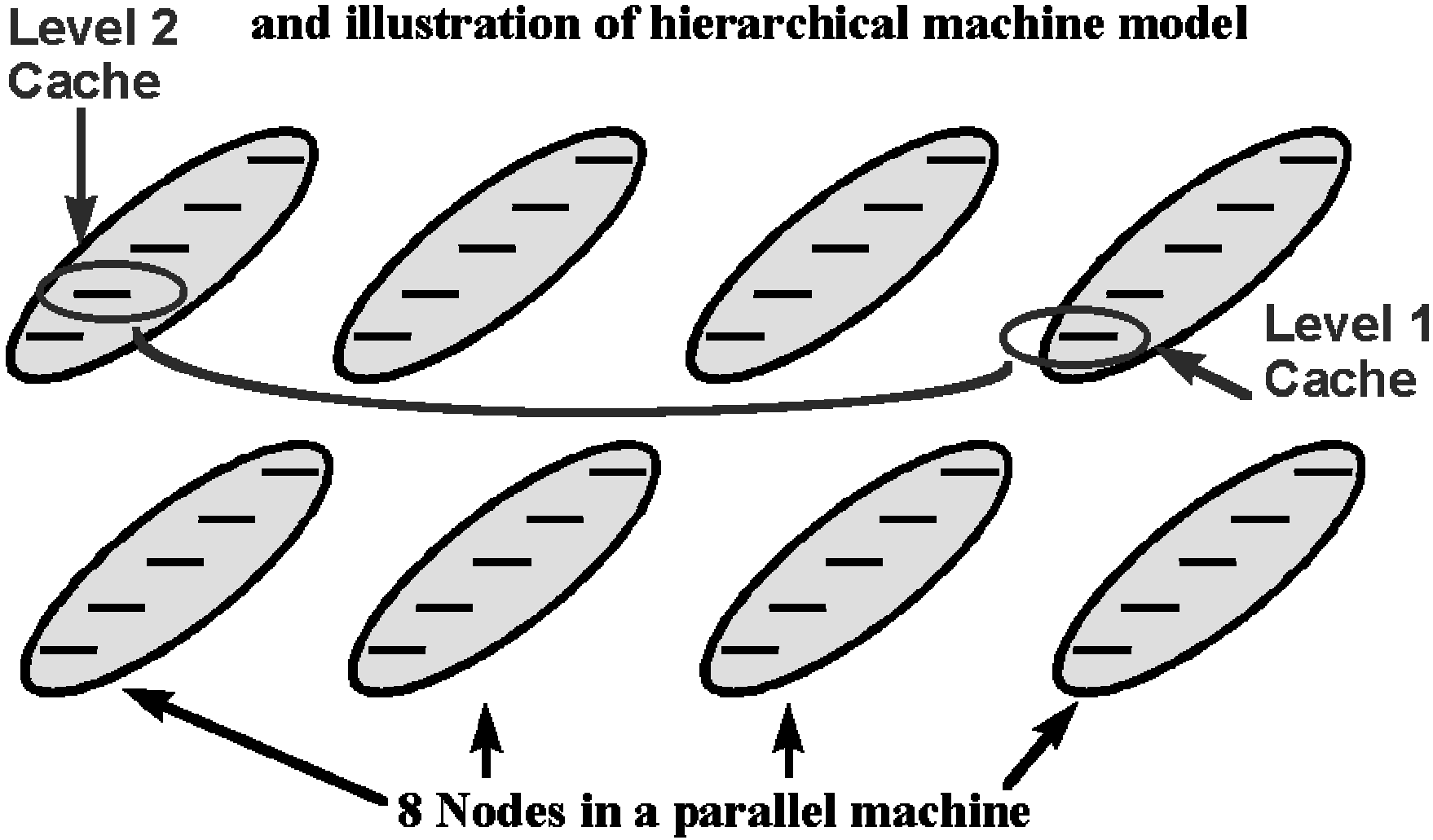
- **Detailed simulations**


Analyse and Classify Existing real Applications
Choose Exemplars of Key Application Classes
Exposing Coarse Grain Module and Data Parallel Structure
Develop Application Emulators in Parallel Fortran C and Java
Use Graphical Interface to design HLAM
1:High Level Representation of Application
2:Target Machine Specification
Transformation/ Runtime Compilation Using Emulated Libraries
3:Execution Script
4:Cost Model
Correct/Improve Representation HLAM
PetaSIM to simulate (sequential/parallel)
Selected Sophisticated Simulation
Validate
Estimated Performance


Full Heterogeneous MetaProblem
Module
MetaProblem
Module
Components
Module
Module
Task Parallelism
Module
Can define multiple levels of Modules
Aggregate
Aggregate
Data Parallelism
Phases of Loosely Synchronous Computation
Splitting of Aggregates to lower levels of Memory Hierarchy

Data Movement in a 5 level Memory Hierarchy

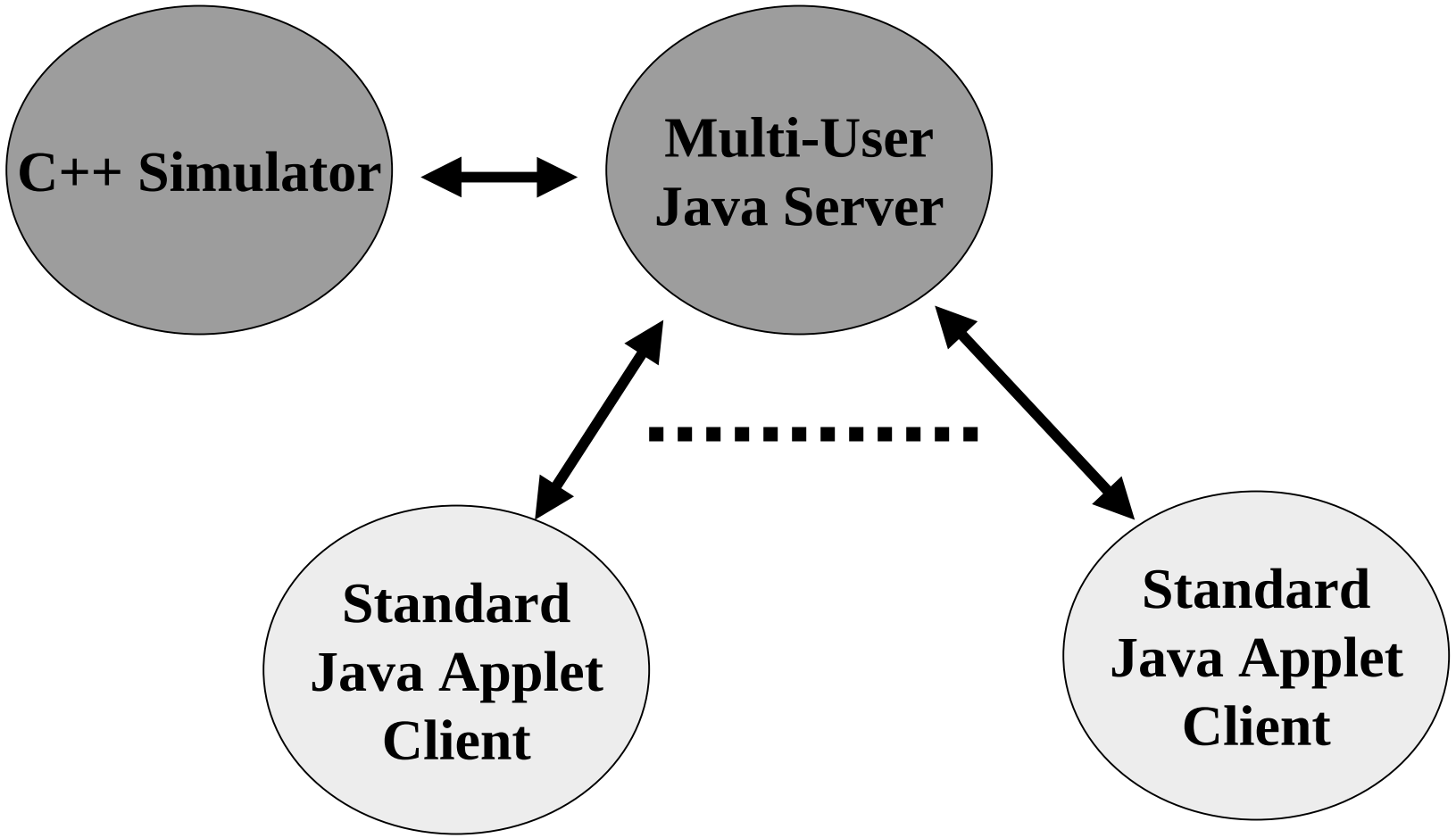
and illustration of hierarchical machine model



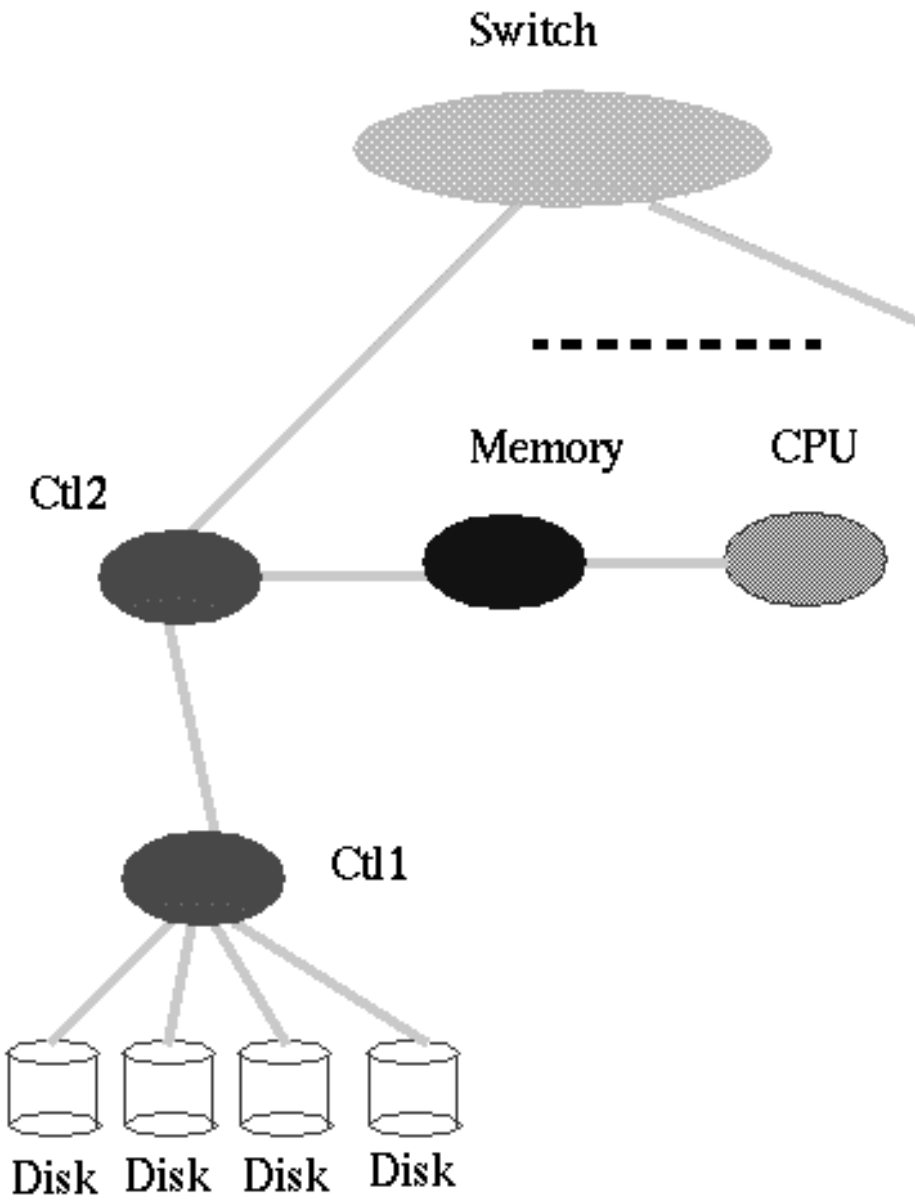
PetaSIM Design

- **We define an object structure for computer (including network) and data**
- **These object representations can be used in dynamic tools such as scheduling (cf. Legion or Globus) and seamless interfaces**
- **Architecture Description**
  - ***nodeset* & *linkset***
  - **(describe the architecture memory hierarchy)**
- **Data Description**
  - ***dataset* & *distribution*** (not stressed in current version)
- **Application Description --** needs further refinement
  - ***execution script***

Architecture of PetaSIM



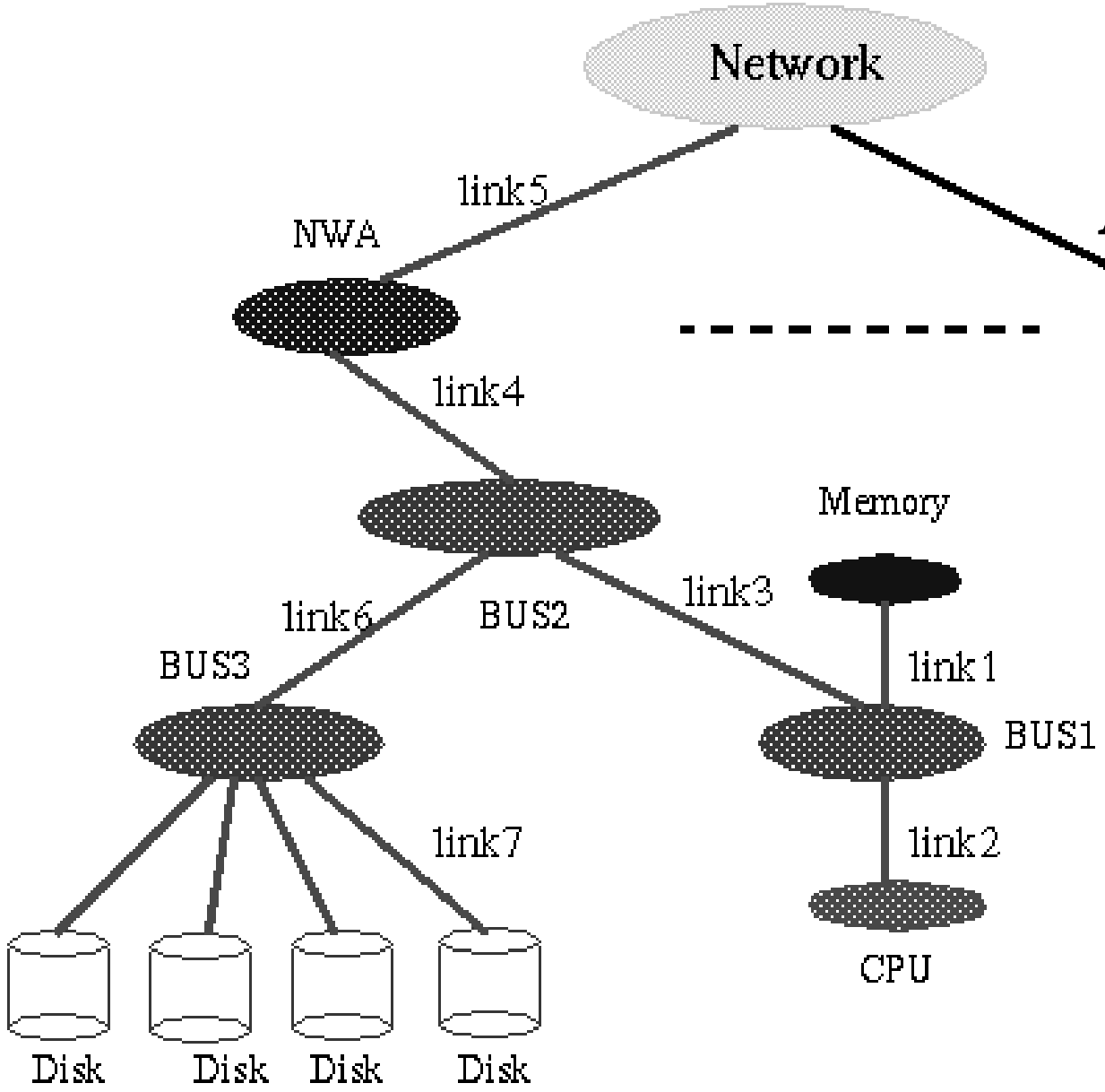
IBM SP2 Architecture I used in tests



SP2 Node Processor Architecture

View as a bunch of nodesets joined by a bunch of linksets
Each component defined as "objects" which are valuable outside PetaSIM

IBM SP2 Architecture II used in tests



SP2 Node Processor Architecture

Nodeset Object Structure

- **Name: one per *nodeset* object**
- **type: choose from memory, cache, disk, CPU, pathway**
- **number: number of members of this *nodeset* in the architecture**
- **grainsize: size in bytes of each member of this *nodeset* (for** memory, cache, disk**)**
- **bandwidth: maximum bandwidth allowed in any one member of this *nodeset***
- **floatspeed: CPU's float calculating speed**
- **calculate(): method used by CPU *nodeset* to perform computation**
- **cacherule: controls persistence of data in a memory or cache**
- **portcount: number of ports on each member of *nodeset***
- **portname[]: ports connected to *linkset***
- **portlink[]: name of *linkset* connecting to this port**
- **nodeset_member_list: list of *nodeset* members in this *nodeset* (for *nodeset* member identification)**

Linkset Object Structure

- **Name: one per *linkset* object**
- **type: choose from updown, across**
- **nodesetbegin: name of initial *nodeset* joined by this *linkset***
- **nodesetend: name of final *nodeset* joined buy this *linkset***
- **topology: used for across networks to specify linkage between members of a single nodeset**
- **duplex: choose from** full or half
- **number: number of members of this *linkset* in the architecture**
- **latency: time to send zero length message across any member of *linkset***
- **bandwidth: maximum bandwidth allowed in any link of this *linkset***
- **send(): method that calculates cost of sending a message across the *linkset***
- **distribution: name of geometric distribution controlling this *linkset***
- **linkset_member_list: list of *linkset* members in this *linkset* (for *linkset* member identification)**

Distribution Object Structure

- **Name: one per distribution object**
- **type: choose from *block1dim, block2dim, block3dim***
  - **Obviously will add more choices here!**

Execution Script

- **Currently a few primitives which stress (unlike most languages) movement of data through hierarchies**
- **send DATAFAMILY from MEM-LEVEL-L to MEM-LEVEL-K**
  - **These reference object names for data and memory *nodesets***
- **move DATAFAMILY from MEM-LEVEL-L to MEM-LEVEL-K**
- **Use distribution DISTRIBUTION from MEM-LEVEL-L to MEM-LEVEL-K**
- **compute DATAFAMILY-A, DATAFAMILY-B,… on MEM-LEVEL-L**
- **synchronize (synchronizes all processors --- loosely synchronous barrier)**

Dataset Object Structure

- **Name: one per *dataset* object**
- **choose from** grid1dim, grid2dim, grid3dim**, specifies type of *dataset***
- **bytesperunit: number of bytes in each unit**
- **floatsperunit: update cost as a floating point arithmetic count**
- **operationsperunit: operations in each unit**
- **update(): method that updates given *dataset* which is contained in a CPU nodeset and a *grainsize* controlled by last memory *nodeset* visited**
- **transmit(): method that calculates cost of transmission of *dataset* between memory levels either communication or movement up and down hierarchy**
  - **Methods can use other parameters or be custom**

Jacobi Example -- Nodeset I

name Type number grainsize portlink (floatspeed)
nodeset_name linkset_name (replicated #times = # links for nodeset in first line)
nodeset_member link_number nodeset_member linkset_member (pair replicated again)

cpu CPU 8 32 1 1.56116e-7
mem link1
cpu0 1 mem0 link10
cpu1 1 mem1 link11
cpu2 1 mem2 link12
cpu3 1 mem3 link13
cpu4 1 mem4 link14
cpu5 1 mem5 link15
cpu6 1 mem6 link16
cpu7 1 mem7 link17

mem Memory 8 134217728 2
cpu link1 ctl2 link2
mem0 2 cpu0 link10 ctl20 link20
mem1 2 cpu1 link11 ctl21 link21
mem2 2 cpu2 link12 ctl22 link22
mem3 2 cpu3 link13 ctl23 link23
mem4 2 cpu4 link14 ctl24 link24
mem5 2 cpu5 link15 ctl25 link25
mem6 2 cpu6 link16 ctl26 link26
mem7 2 cpu7 link17 ctl27 link27

disks Disk 8 2147483648 1
ctl1 link3
d0 1 ctl10 link30
d1 1 ctl11 link31
d2 1 ctl12 link32
d3 1 ctl13 link33
d4 1 ctl14 link34
d5 1 ctl15 link35
d6 1 ctl16 link35
d7 1 ctl17 link37

Jacobi Example -- Nodeset II

```
ctl1 Pathway 8 0 2
disks link3 ctl2 link4
ctl10 2 d0 link30 ctl20 link40
ctl11 2 d1 link31 ctl21 link41
ctl12 2 d2 link32 ctl22 link42
ctl13 2 d3 link33 ctl23 link43
ctl14 2 d4 link34 ctl24 link44
ctl15 2 d5 link35 ctl25 link45
ctl16 2 d6 link36 ctl26 link46
ctl17 2 d7 link37 ctl27 link47
```

```
ctl2 Pathway 8 0 3
mem link2 ctl1 link4 network link5
ctl20 3 mem0 link20 ctl10 link40 network0 link50
ctl21 3 mem1 link21 ctl11 link41 network0 link51
ctl22 3 mem2 link22 ctl12 link42 network0 link52
ctl23 3 mem3 link23 ctl13 link43 network0 link53
ctl24 3 mem4 link24 ctl14 link44 network0 link54
ctl25 3 mem5 link25 ctl15 link45 network0 link55
ctl26 3 mem6 link26 ctl16 link46 network0 link56
ctl27 3 mem7 link27 ctl17 link47 network0 link57

network Switch 1 0 1
ctl2 link5
network0 8 ctl20 link50 ctl21 link51 ctl22 link52
ctl23 link53 ctl24 link54 ctl25 link55 ctl26 link56
ctl27 link57
```

Jacobi Example -- Linkset I

```
link1 Updown Full 8 0.0 524288000
mem cpu
link10 mem0 cpu0
link11 mem1 cpu1
link12 mem2 cpu2
link13 mem3 cpu3
link14 mem4 cpu4
link15 mem5 cpu5
link16 mem6 cpu6
link17 mem7 cpu7

link2 Updown Full 8 0.0 83886080
mem ctl2
link20 mem0 ctl20
link21 mem1 ctl21
link22 mem2 ctl22
link23 mem3 ctl23
link24 mem4 ctl24
link25 mem5 ctl25
link26 mem6 ctl26
link27 mem7 ctl27
```

Excerpt from Linkset Definitions

1)name Type Duplex number latency bandwidth

2)nodesetbegin nodesetend

3)linkset_member nodeset_member_begin nodeset_member_end

```
link3 Updown Full 8 2.0e-4 8388608
ctl1 disks
link30 ctl10 d0
link31 ctl11 d1
link32 ctl12 d2
link33 ctl13 d3
link34 ctl14 d4
link35 ctl15 d5
link36 ctl16 d6
link37 ctl17 d7
```

Jacobi Example -- Linkset II

- **Excerpt from Linkset Definitions:**
- **name Type Duplex number latency bandwidth**
- **nodesetbegin nodesetend**
- **linkset_member nodeset_member_begin nodeset_member_end**

```
link4 Updown Full 8 0.0 83886080
ctl1 ctl2
link40 ctl10 ctl20
link41 ctl11 ctl21
link42 ctl12 ctl22
link43 ctl13 ctl23
link44 ctl14 ctl24
link45 ctl15 ctl25
link46 ctl16 ctl26
link47 ctl17 ctl27
```

```
link5 Updown Full 8 4.0e-5 83886080
ctl2 network
link50 ctl20 network0
link51 ctl21 network0
link52 ctl22 network0
link53 ctl23 network0
link54 ctl24 network0
link55 ctl25 network0
link56 ctl26 network0
link57 ctl27 network0
```

Jacobi Example

Dataset Definition is very simple (by design):

name type size bytesperunit floatperunit operationperunit

Jacobi Grid2dim 1000000 4 4 10

Jacobi Example -- Execution Script I

compute Jacobi on cpu4
Move Jacobi from d1 to ctl11
move Jacobi from ctl11 to ctl21
move Jacobi from ctl21 to mem1
move Jacobi from mem1 to ctl21
move Jacobi from ctl21 to network0
move Jacobi from network0 to ctl20
move Jacobi from ctl20 to mem0
move Jacobi from mem0 to cpu0
compute Jacobi on cpu0
move Jacobi from mem1 to ctl21
move Jacobi from ctl21 to network0
move Jacobi from network0 to ctl22
move Jacobi from ctl22 to mem2
move Jacobi from mem2 to cpu2
compute Jacobi on cpu2
move Jacobi from mem1 to ctl21
move Jacobi from ctl21 to network0
move Jacobi from network0 to ctl25
move Jacobi from ctl25 to mem5
move Jacobi from mem5 to cpu5
compute Jacobi on cpu5

Move Jacobi from d0 to ctl10
move Jacobi from ctl10 to ctl20
move Jacobi from ctl20 to mem0
move Jacobi from mem0 to ctl20
move Jacobi from ctl20 to network0
move Jacobi from network0 to ctl21
move Jacobi from ctl21 to mem1
move Jacobi from mem1 to cpu1
compute Jacobi on cpu1
move Jacobi from mem0 to ctl20
move Jacobi from ctl20 to network0
move Jacobi from network0 to ctl24
move Jacobi from ctl24 to mem4
move Jacobi from mem4 to cpu4

Also a simpler data parallel version

Jacobi Example -- Execution Script II

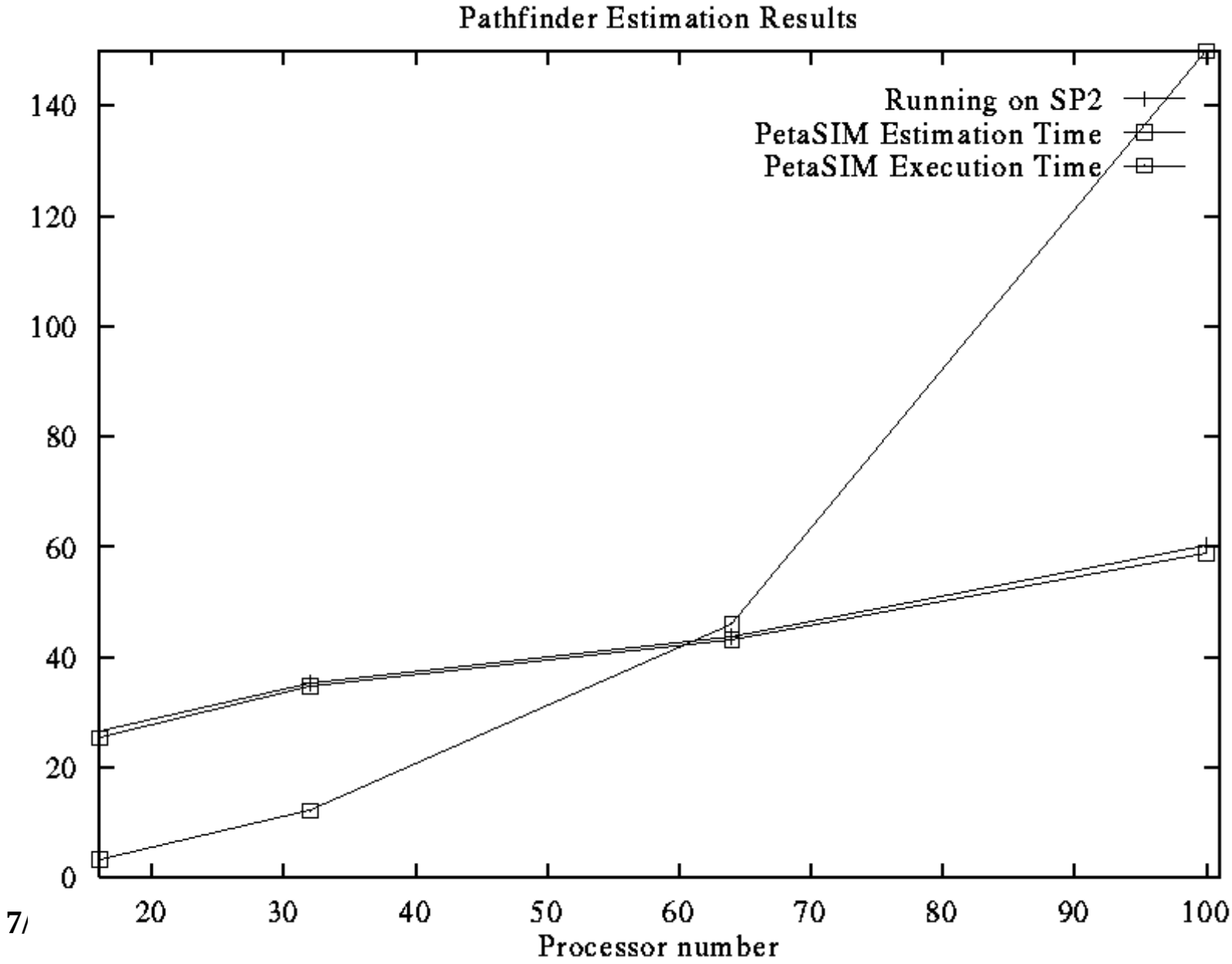
```
Move Jacobi from d5 to ctl15
move Jacobi from ctl15 to ctl25
move Jacobi from ctl25 to mem5
move Jacobi from mem5 to ctl25
move Jacobi from ctl25 to network0
move Jacobi from network0 to ctl21
move Jacobi from ctl21 to mem1
move Jacobi from mem1 to cpu1
compute Jacobi on cpu1
move Jacobi from mem6 to ctl26
move Jacobi from ctl26 to network0
move Jacobi from network0 to ctl27
move Jacobi from ctl27 to mem7
move Jacobi from mem7 to cpu7
compute Jacobi on cpu7
```

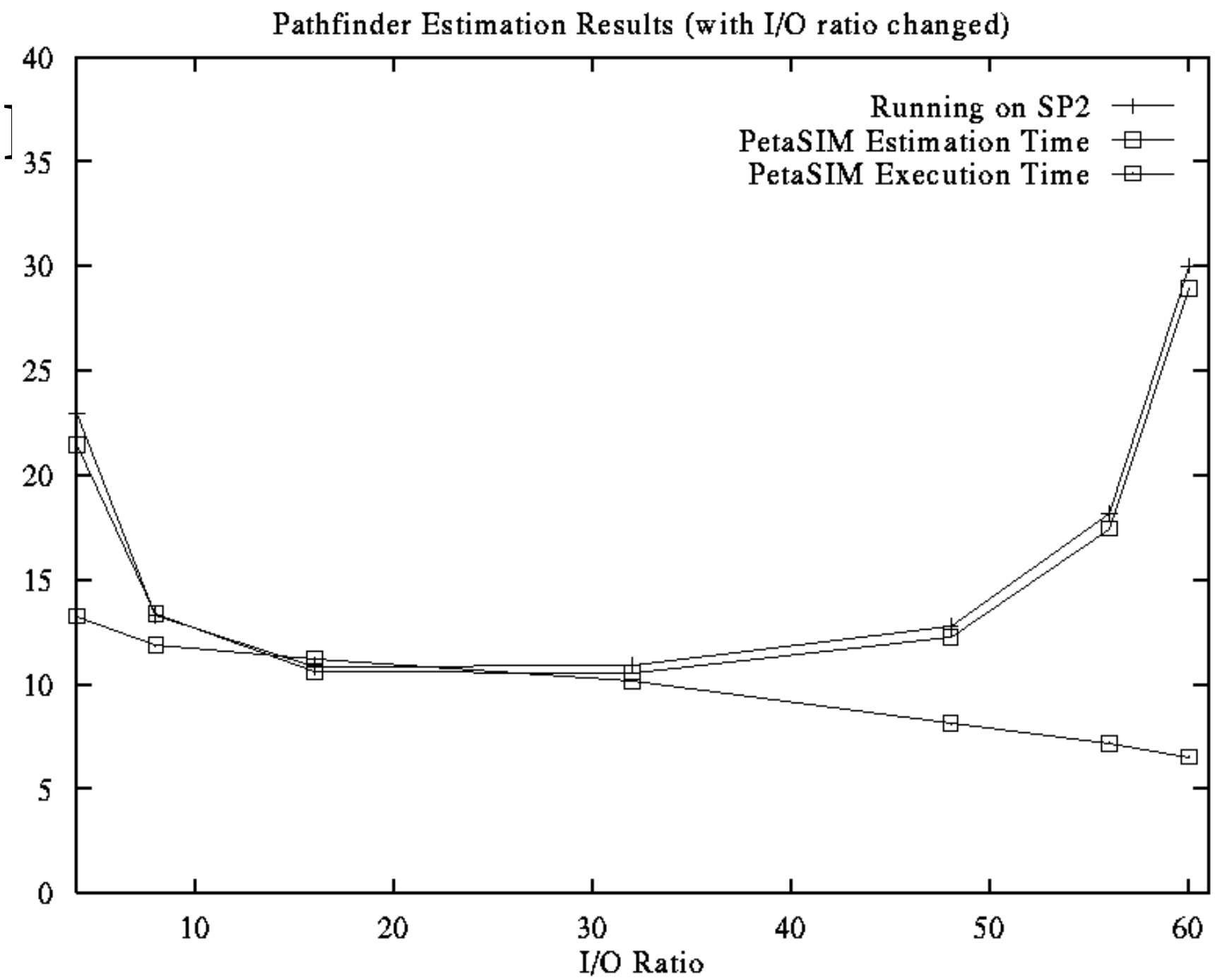
```
Move Jacobi from d7 to ctl17
move Jacobi from ctl17 to ctl27
move Jacobi from ctl27 to mem7
move Jacobi from mem7 to ctl27
move Jacobi from ctl27 to network0
move Jacobi from network0 to ctl23
move Jacobi from ctl23 to mem3
move Jacobi from mem3 to cpu3
compute Jacobi on cpu3
move Jacobi from mem7 to ctl27
move Jacobi from ctl27 to network0
move Jacobi from network0 to ctl26
move Jacobi from ctl26 to mem6
move Jacobi from mem6 to cpu6
compute Jacobi on cpu6
synchronize
```

Pathfinder Performance Estimation Results (Architecture II)

Examples	IBM SP2 Running Time	Petasim Estimation Time(appl)	Petasim Execution Time(appl)	Petasim Estimation Time(max)	Petasim Execution Time(max)
proc16path	26.48	25.38	3.41	21.12	3.4
proc32path	35.38	34.68	12.18	29.11	12.24
proc64path	43.80	43.07	46.01	36.86	45.95
proc100path	60.34	58.89	149.91	52.68	149.87
path.vary(4_60)	22.99	21.46	13.26	15.45	13.26
path.vary(8_56)	13.32	13.37	11.86	10.08	11.85
path.vary(16_48)	10.85	10.63	11.24	9.12	11.25
path.vary(32_32)	10.93	10.57	10.15	9.89	10.18
path.vary(48_16)	12.79	12.28	8.13	11.84	8.14
path.vary(56_8)	18.20	17.45	7.19	17.18	7.20
path.vary(60_4)	30.02	28.93	6.52	28.68	6.47

Pathfinder Performance Estimation Results




Pathfinder Estimation Results (with I/O ratio changed)
Running on SP2
PetaSIM Estimation Time
PetaSIM Execution Time
40
35
30
25
20
15
10
5
0
10
20
30
40
50
60
I/O Ratio

PetaSIM Estimation Results (Architecture I)

Examples	IBM SP2 Running Time	Petasim Estimation Time	Petasim Execution Time
proc15titan(9K)	114	105.23	7.12
proc15titan(27K)	347	318.51	21.61
proc32titan(27K)	N/A	316.07	41.69
proc15path(9K)	166	159.45	8.49
proc15path(27K)	497	484.51	25.7
proc32path(55K)	N/A	710.48	78.99

Interface between Emulator and PetaSIM Part of Pathfinder Application

Excerpt from Nodeset Definitions

name Type number grainsize portlink (floatspeed)

nodeset_name linkset_name

nodeset_member link_number nodeset_member linkset_member

cpu CPU 8 32 1 1.530000e-06
bus1 link2
cpu0 1 bus10 link20
cpu1 1 bus11 link21
cpu2 1 bus12 link22
cpu3 1 bus13 link23
cpu4 1 bus14 link24
cpu5 1 bus15 link25
cpu6 1 bus16 link26
cpu7 1 bus17 link27
…….

mem Memory 8 134217728 1
bus1 link1
mem0 1 bus10 link10
mem1 1 bus11 link11
mem2 1 bus12 link12
mem3 1 bus13 link13
mem4 1 bus14 link14
mem5 1 bus15 link15
mem6 1 bus16 link16
mem7 1 bus17 link17

Interface between Emulator and PetaSIM -- Pathfinder

Excerpt from Linkset Definitions:

name Type Duplex number latency bandwidth

nodesetbegin nodesetend

linkset_member nodeset_member_begin nodeset_member_end

link1 Updown Full 8 0.000000e+00 134217728
bus1 mem
link10 bus10 mem0
link11 bus11 mem1
link12 bus12 mem2
link13 bus13 mem3
link14 bus14 mem4
link15 bus15 mem5
link16 bus16 mem6
link17 bus17 mem7

link2 Updown Full 8 0.000000e+00
134217728
bus1 cpu
link20 bus10 cpu0
link21 bus11 cpu1
link22 bus12 cpu2
link23 bus13 cpu3
link24 bus14 cpu4
link25 bus15 cpu5
link26 bus16 cpu6
link27 bus17 cpu7

Interface between Emulator and PetaSIM -- Pathfinder

Excerpt from Dataset Definitions:

name type size bytesperunit floatperunit operationperunit

data0 grid2dim 46920 4 1 1
data1 grid2dim 46920 4 1 1
data2 grid2dim 46920 4 1 1
data3 grid2dim 46920 4 1 1
data4 grid2dim 46920 4 1 1
data5 grid2dim 46920 4 1 1
data6 grid2dim 46920 4 1 1
data7 grid2dim 46920 4 1 1
data8 grid2dim 46920 4 1 1
data9 grid2dim 46920 4 1 1
data10 grid2dim 46920 4 1 1
data11 grid2dim 46920 4 1 1
data12 grid2dim 46920 4 1 1
data13 grid2dim 46920 4 1 1
data14 grid2dim 46920 4 1 1
……...

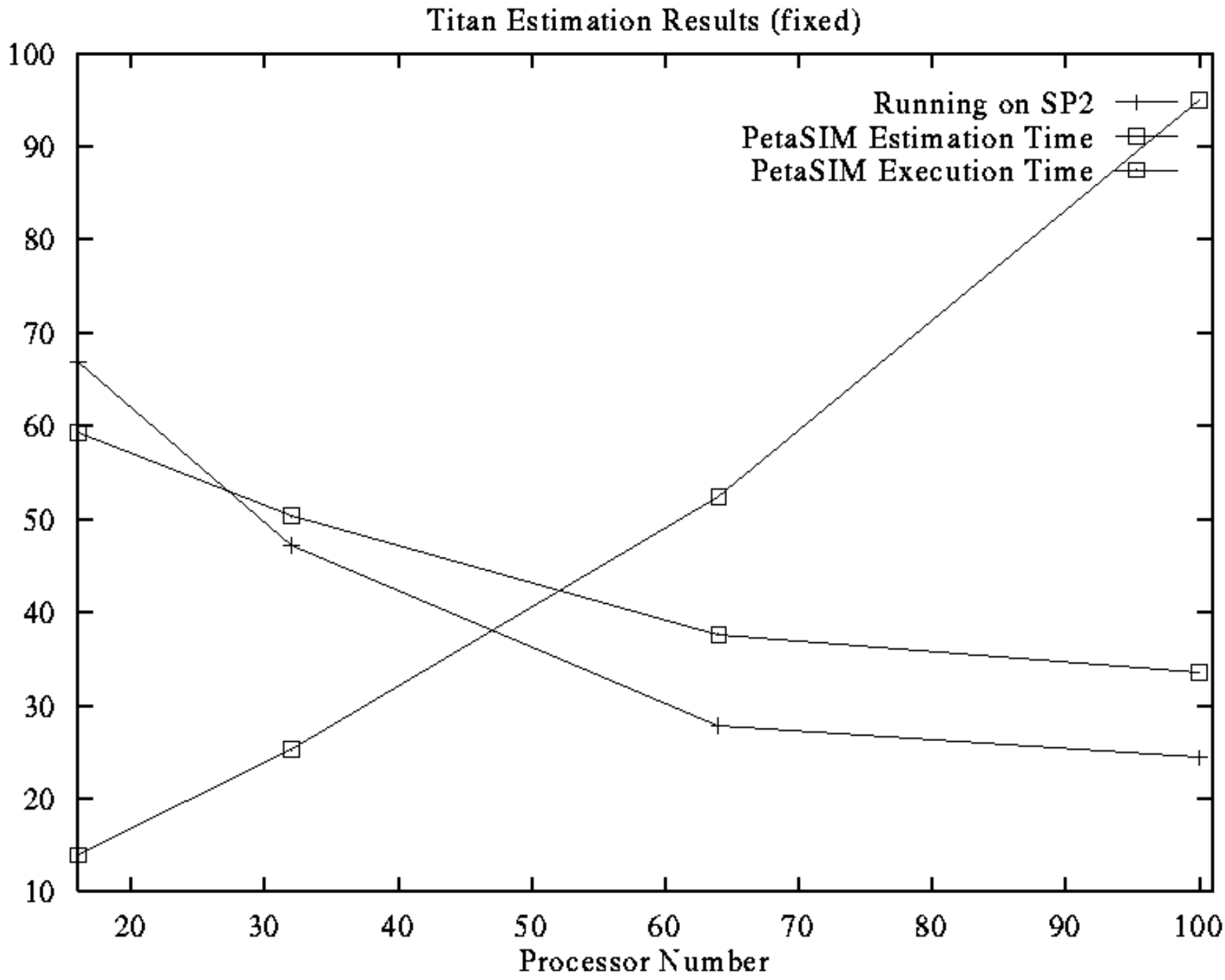
Interface between Emulator and PetaSIM Pathfinder

Excerpt from Execution Script:

```
move data0 from disks0 to bus30
move data0 from bus30 to bus20
move data0 from bus20 to bus10
move data0 from bus10 to mem0
move data0 from mem0 to bus10
move data0 from bus10 to bus20
move data0 from bus20 to nwa0
move data0 from nwa0 to network0
move data0 from network0 to nwa11
move data0 from nwa11 to bus211
move data0 from bus211 to bus111
move data0 from bus111 to mem11
move data0 from mem11 to bus111
move data0 from bus111 to cpu11
compute data0 on cpu11
```

Titan Estimation Results (Architecture II)

Examples	IBM SP2 Running Time	Petasim Estimation Time(appl)	Petasim Execution Time(appl)	Petasim Estimation Time(max)	Petasim Execution Time(max)
proc16titan	66.51	59.95	13.89	59.52	13.84
proc32titan	94.16	101.30	50.65	100.48	50.43
proc64titan	112.42	156.52	209.44	154.85	209.63
proc100titan	154.90	226.56	594.49	224.37	595.71
proc16fixed	66.83	59.95	13.93	59.52	13.82
proc32fixed	47.16	50.32	25.24	49.92	25.18
proc64fixed	27.84	37.63	52.37	37.26	52.23
proc100fixed	24.37	33.56	94.98	33.31	94.50


Titan Estimation Results (fixed)
Running on SP2
PetaSIM Estimation Time
PetaSIM Execution Time
100
90
80
70
60
50
40
30
20
10
20
30
40
50
60
70
80
90
100
Processor Number

VMScope Performance Estimation Results (Architecture II)

Examples	IBM SP2 Running Time	Petasim Estimation Time(appl)	Petasim Execution Time(appl)	Petasim Estimation Time(max)	Petasim Execution Time(max)
proc16vmscope	17.13	14.59	0.25	14.58	0.26
proc32vmscope	21.35	17.80	0.67	17.79	0.68
proc64vmscope	21.43	18.02	2.11	18.00	2.12
proc100vmscope	18.50	15.45	4.76	15.43	4.84

PetaSIM Current Progress Summary

- **Architecture Description (*nodeset* & *linkset*)**
- **Application Description (*dataset* & execution script)**
- **Link to Application Emulators**
- **Jacobi hand-written example**
- **Pathfinder, Titan, VMScope real applications (Generated by UMD's Emulator)**
- **Easy modified Architecture and Application description**
- **Fast and relatively Accurate performance estimation (PetaSIM running on single processor)**
- **Java applet based user Interface**
- **About 5000 lines of C++ and 4000 lines of Java (client and server)**

Possible Future Work

- **Richer set of applications using standard benchmarks and DoD MSTAR**
- **Relate object model to those used in "seamless interfaces" / metacomputing i.e. to efforts to establish (distributed) object model for computation**
- **Review very simple execution script -- should we add more complex primitives or regard "application emulators" as this complex script**
- **Binary format ("compiled PetaSIM") of architecture and application description (ASCII format will make execution script very large)**
  - **Translation tool from ASCII format to binary format (to retain the friendly user interface)**
- **Upgrade performance evaluation model**
- **Run performance simulation in parallel (i.e. PetaSIM running on multi-processors)**