Modeling Computation for HW/SW Codesign

PACT 2013 Keynote

David J. Kuck
Software & Services Group

Edinburgh
Sept. 9, 2013

Outline

A. Introduction

B. Basic Equations

C. System Codesign Models

D. Measurement

E. Cape Modeling Results

F. Conclusions

Where does HW/SW design stand

- High-tech, successful, well-advanced field
  - Arguably a major technical/business success
- Unlike similar fields: no system equations
  - Formal models: devices, circuits, algorithms, num. anal., …
  - Intuition: System design, SW design, cost/performance, …
- Much work on "getting the right balance"
  - What is "right balance," how is it defined?
  - How do we get there from here?
  - There is no foundation of global procedures.
- There are several ways to study these gaps
  - How to use computers better to design computer systems?

Goal: HW/SW Codesign Methodology that handles all of this

3 Steps from Basic Codesign Theory

- Start with a basic codesign theory
  - See codesign section; then →

1. Build a *model+optimization* tool: solve toy problems
   - *Done and validated* on minimal real data
   - Fast simulation or constrained optimal system design
2. *Measure* real machines
   - SW/HW measurement of many nodes *in process*
     - Vector access and processing
     - Memory hierarchy levels
     - Parallelism, throughput, clock freq, power, energy
3. *Model* new HW/arch ideas
     - Optimize BW of all nodes for perf/energy
     - Vector, parallel tradeoffs
     - BW and size of cache needed? (e.g. L2=0 ?)

Modeling and Measurement Steps

- Choose HW *nodes* to match design needs
- Choose SW *phases* with steady-state behavior
- Recognize individually, understand joint behavior

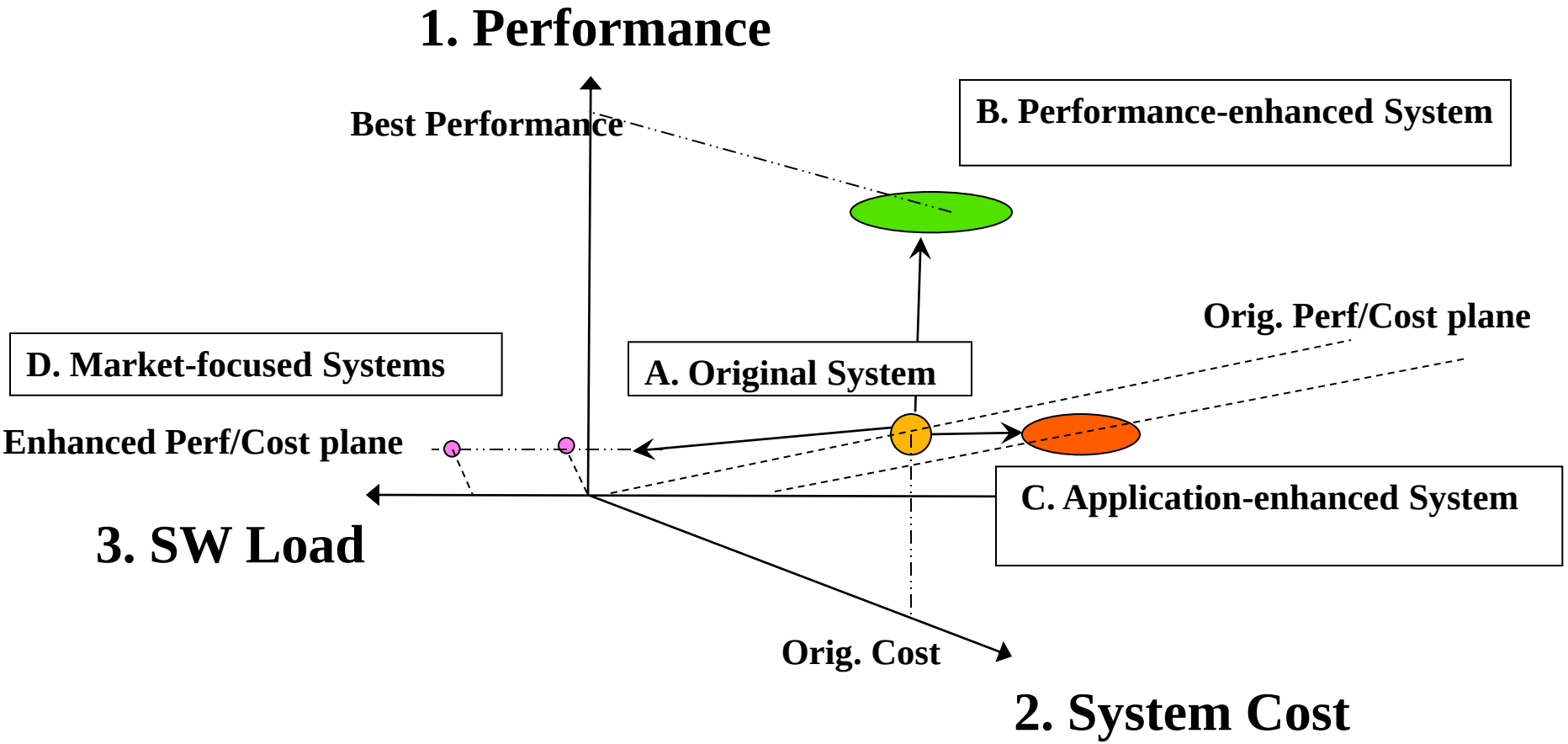
- The above *enables* measurement and modeling
  - Physical system reconstruction from modeling results
- Captures details, supports optimization – Section C.

- Measurement is a challenge – Section D.
  - HW counters are weak, SW methods have been weaker

B. Basic Equations

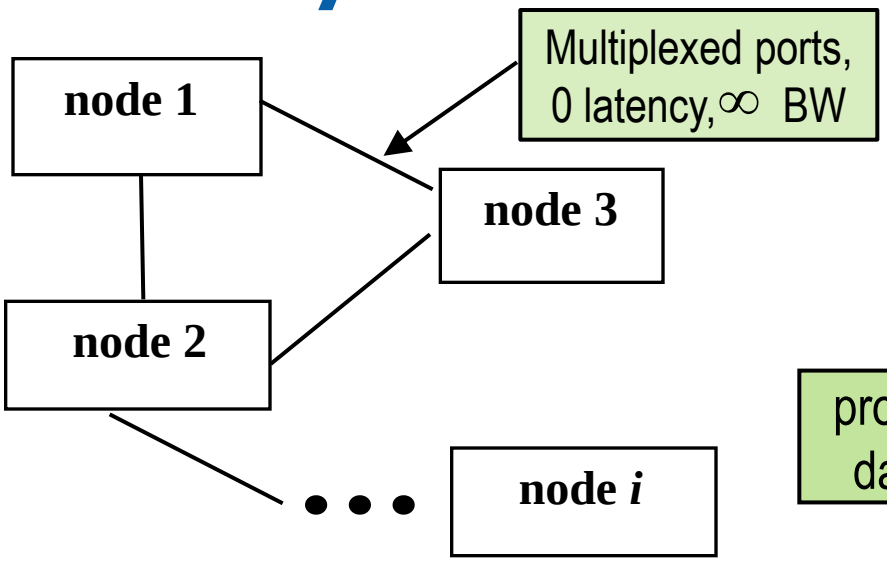
B.1. Modeling: 4d Codesign Space

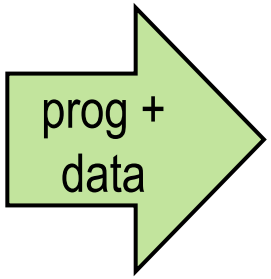
(w.o. Oper. Cost E)

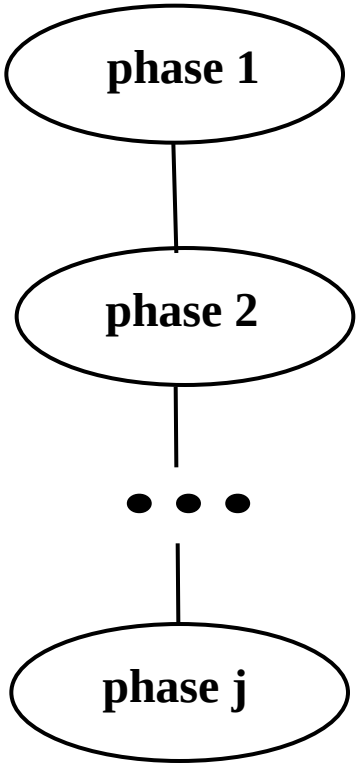


Electrical dimension: energy via (clock *f*, *V*) → Operating Cost

2 Models: Prog+Data+HWsyst→
a. HW System Arch. b. SW Computation







Node *i* has ***Bandwidth*** $= B_i \left[\frac{b;O}{s} \right]$,

BW ratio= $\alpha_{i,j} = B_j / B_i$

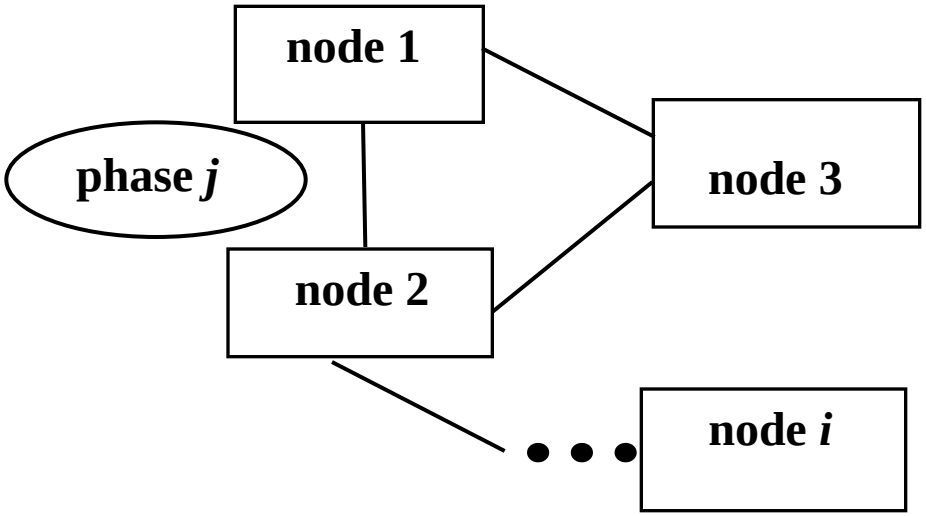
Power $= W_i[w]$,

Energy/bit $= \gamma_i \left[\frac{w*s}{b} \right]$

Phase *j* has ***Operation count*** $= O_{i,j}[ops]$ on node *i*

Physical model characteristics

A Computation on a System Produces:



For node *i*, phase *j* :

- ***Running time*** = $t_{i,j}[s]$
- ***Energy*** = $E_{i,j}[j \text{ or } w \cdot hr]$
- ***Computational Capacity*** = $C_{i,j}\left[\frac{b;O}{s}\right]$
- ***Relative capacity saturation:***

$$0 \le \sigma_i^C = C_i / B_i \le 1$$

- ***Capacity Equation*** (node *i*):

$$\begin{cases} C_{i,j} = O_{i,j} / t_{i,j} \\ C_{i,j} = \overline{B}_i, \quad i \text{ saturated} \end{cases}$$

- ***Capacity Intensity Equation*** (nodes *i,k*): $C_{i,j} = \mu_{ki,j} C_{k,j}$

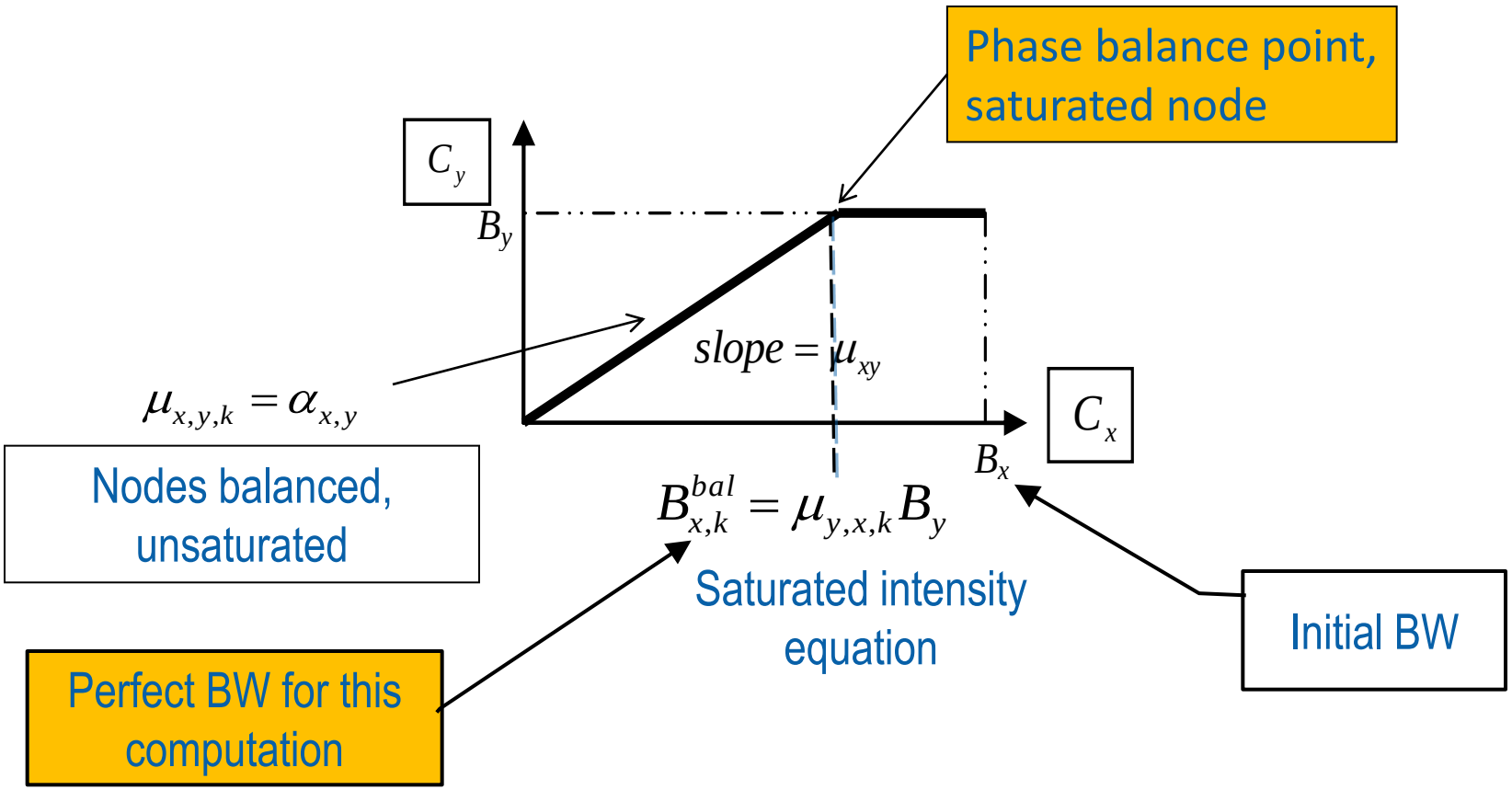
Each phase saturates one or more nodes

Computational codesign variable relations

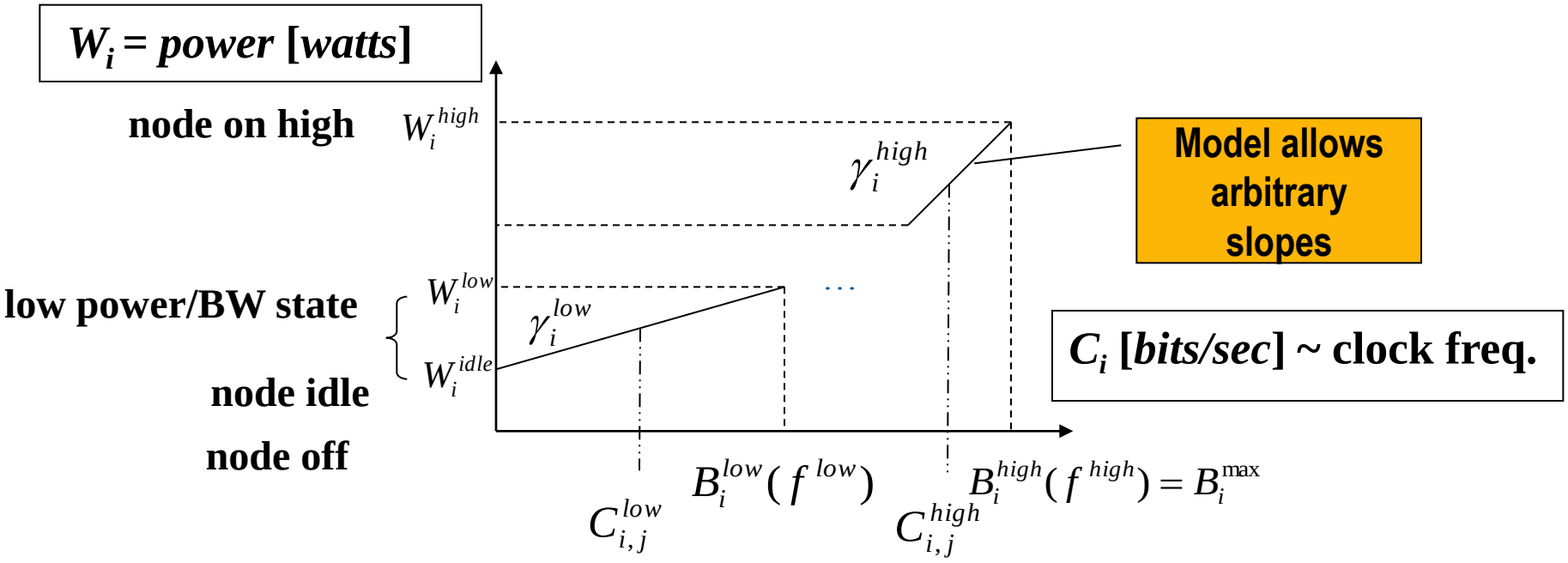
- Independent variables of computation
  - Time (t), HW speed (f,V) abb. (f), SW ops ($O(D)$) *t, ops*
- Basic dependent variables
  - HW: BW $B(f)$, power $W^{\max}(f), W^{idle}(f), W^{dyn}(f)$ *Power*
  - SW/HW: *Computational capacity* $C \le B$ *Speed*
- Derived dependent variables (mostly HW/SW dependent)
  - HW: *Energy/bit* = $\gamma_i \left[\frac{w * s}{b} \right]$
  - *Energy: E*
  - *Relative C saturation*: $0 \le \sigma_i^C = C_i / B_i \le 1$
  - *Intensity*: $C_{i,j} = \mu_{ki,j} C_{k,j}$
  - *Quality*: *E/C*

Want linear models, but deal with nonlinearity

Balanced nodes, phase balance point



B.3. W and E: Average-power model



$\gamma_{i,j}^{k}$ [E/b] for node i, phase j, power state *k*, $1 \le k \le s$

Power Equation $W_{i,j}^{k} = \gamma_{i,j}^{k} C_{i,j}^{k} + W_i^{kidle}$

Combine with architecture codesign model

B.4. Modeling Summary: Node Property Eqs.

Capacity $0 \le C_{i,j} = O_{i,j} / t_{i,j} \le B_i^k(f,V) \le B_i^{\max} = \max_k \{C_{i,k}\} -> B_i^{phy}$

Latency $0 \le B_i^{-1} = L_i^{\min} \le L_i = C_i^{-1} \le L_i^{\max}$ ← Includes cache miss time

Power $0 \le W_i = \gamma_i C_i + W_i^{idle} \le W_i^{\max}$

4 Local Eqs.
node *i*, phase *j*

Energy $E_{i,j} = W_{i,j} t_{i,j} = \gamma_{i,j} O_{i,j} + W_i^{idle} t_{i,j}$

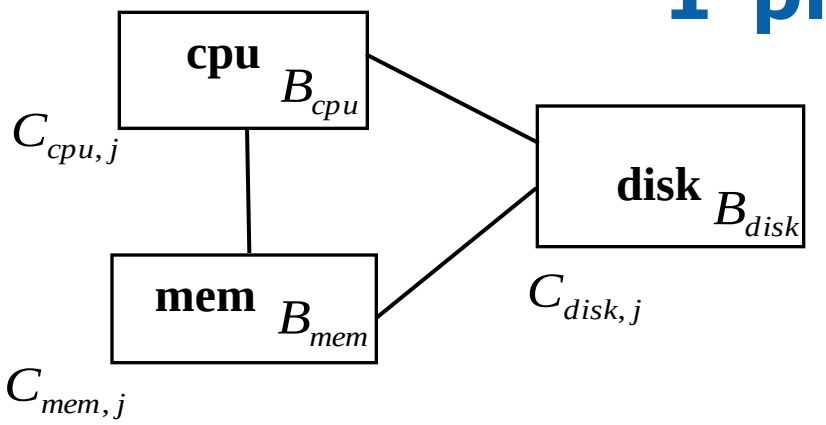
3 Global Eqs.

- **Initial Cost =** $\sum B_j$
- **Performance =** $\sum\sum f(C_{i,j})$
- **Operating Cost =** $\sum\sum E_{i,j}$

above + internode equations = complete set

C. System Codesign Models

B.2. Codesign Examples:
1-phase, 3-node system



For node *i*, phase *j* :

- ***Computational Capacity*** **=** $C_{i,j}\left[\frac{b}{s}\right]$

- ***Capacity Equation*** **(node *i*):** (At least one node is saturated)

$$\begin{cases} C_{i,j} = O_{i,j} / t_{i,j} \\ C_{i,j} = \overline{B}_i \qquad i \text{ saturated} \end{cases}$$

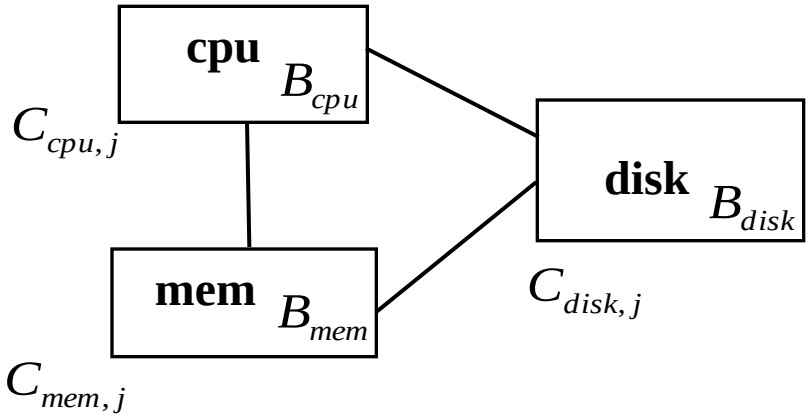
- ***Intensity Equation*** **(nodes *i,k*):** $C_{i,j} = \mu_{ki,j} C_{k,j}$

How do we systematically increase system performance?

- Focus on **saturated** node(s)
- Intensity $\mu_{ki,j}$ is **invariant** if SW is not changed (program and data)

Simple design questions follow

Ex.1: Systematically boost syst. perf.?



$$C_{disk,j} = \overline{B}_{disk} \rightarrow \overline{B}^{fast}_{disk}$$

saturated

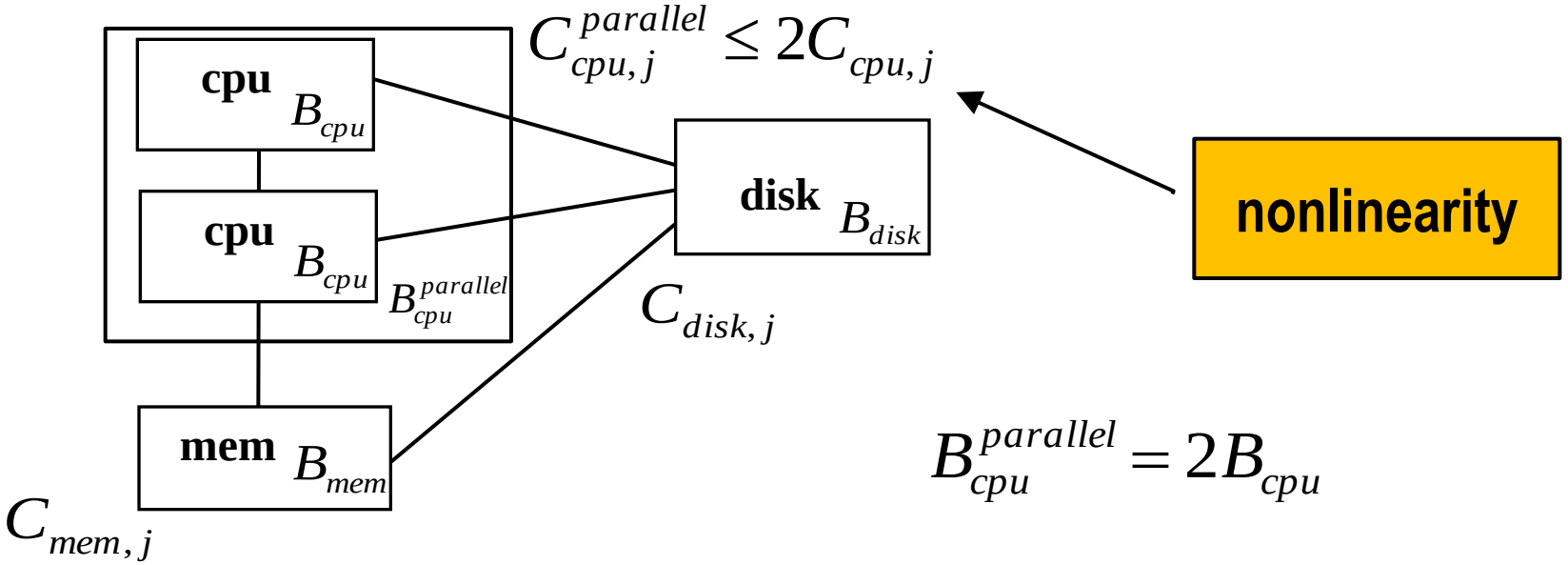
If $C_{disk,j} = \overline{B}_{disk}$, increase *disk performance*:

- **Set saturated node BW to desired performance level $\rightarrow \overline{B}^{fast}_{disk}$**
- **Adjust other nodes accordingly → use $C_{cpu,j} = \mu_{disk-cpu,j}\, C_{disk,j}$**

What if performance demand exceeds the fastest cpu or mem available?

Drive toward $B^{waste} = B - C = 0$, subject to discrete node values

Ex.2: Perf. demand > fastest node available?

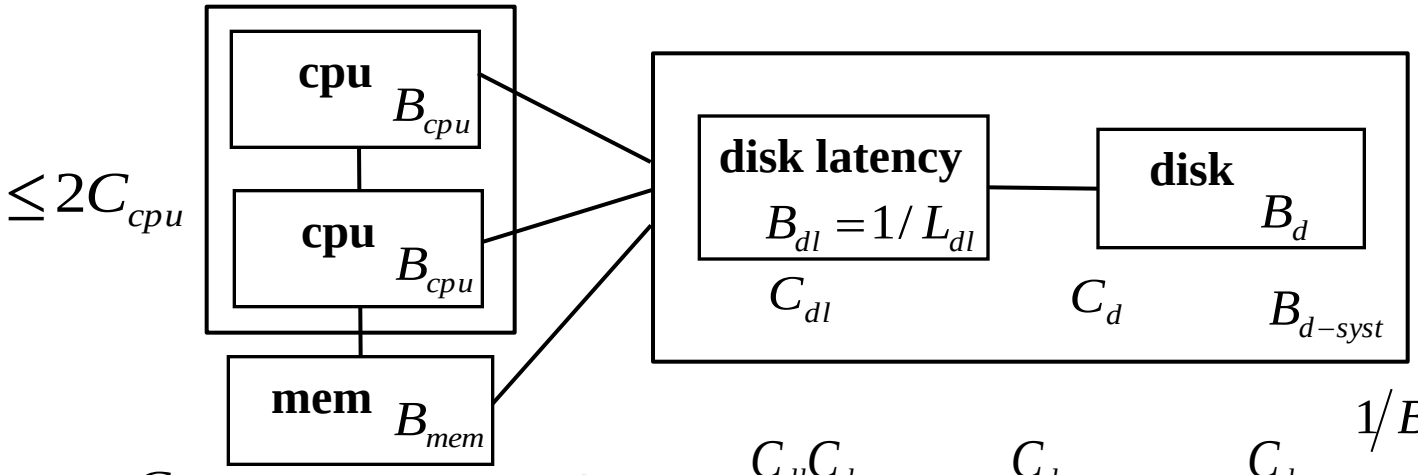


If $C_{cpu,j}^{goal} > B_{cpu}^{\max}$, use *parallel cpu* model:

- **Replicate node – parallel processors or memories add BW**
  - $B_{ik}^{parallel} = B_i + B_k$

Parallel capacities lead to *multirate* nodes, $C_{cpu,j}^{parallel} \le 2C_{cpu,j}$

Ex.3: What if disk has latency?



$$C_{d-syst} = \frac{C_{dl}C_d}{C_{dl}+C_d} = \frac{C_d}{1+C_d/C_{dl}} = \frac{C_d}{1+C_dL_{dl}}$$

$$1/B_{d-syst}^{ser\,non-ov} = 1/B_d + 1/B_{dl}$$

Latency models: transmission, contention, or rotational delay?

- Physical transmission delay – constant, function of wire length
- Serial nodes add reciprocal BWs, nonlinear capacity

$$1/B_{ik}^{ser\,non-ov} = 1/B_i + 1/B_k \qquad B_{ik}^{ser\,ov} = \min\{B_i, B_k\}$$

Variable latencies lead to nonlinear *multirate* nodes

Summary Internode Equations

Local Eqs.

Capacity Intensity : $C_{i,j} = \mu_{ki,j} C_{k,j} \rightarrow \mu_{ki,j} \overline{B}_k$

Parallel supernode $C_{ij}^{parallel} = C_i + C_j$

Serial supernode $1/C_{ij}^{sernon-ov} = 1/C_i + 1/C_j; \quad C_{ij}^{serov} = \min\{C_i, C_j\}$

Multirate nonlinear supernode $C_i = \dfrac{f_1(C)}{1 + f_2(f_3 - 1)} \qquad 0 \le f_2, f_3 \le 1$

Piecewise linearize

fractional use

perf function

System of (nodes X phases) inequalities➔ Global Codesign

Objective Functions

$$\min B_{system}^{waste} = \min \sum_{i=1}^{n} (B_i - \sum_{j=1}^{m} \phi_j C_{i,j})$$

HW

HW/SW

= min *(initial cost) – max (performance)*

$$\min E_{i,j} = \min \sum_{i} \sum_{j} [\gamma_{i,j} O_{i,j} + W_i^{idle} t_{i,j}]$$

SW

HW/SW

HW

$$\min \frac{E_{system}}{C_{system}}$$

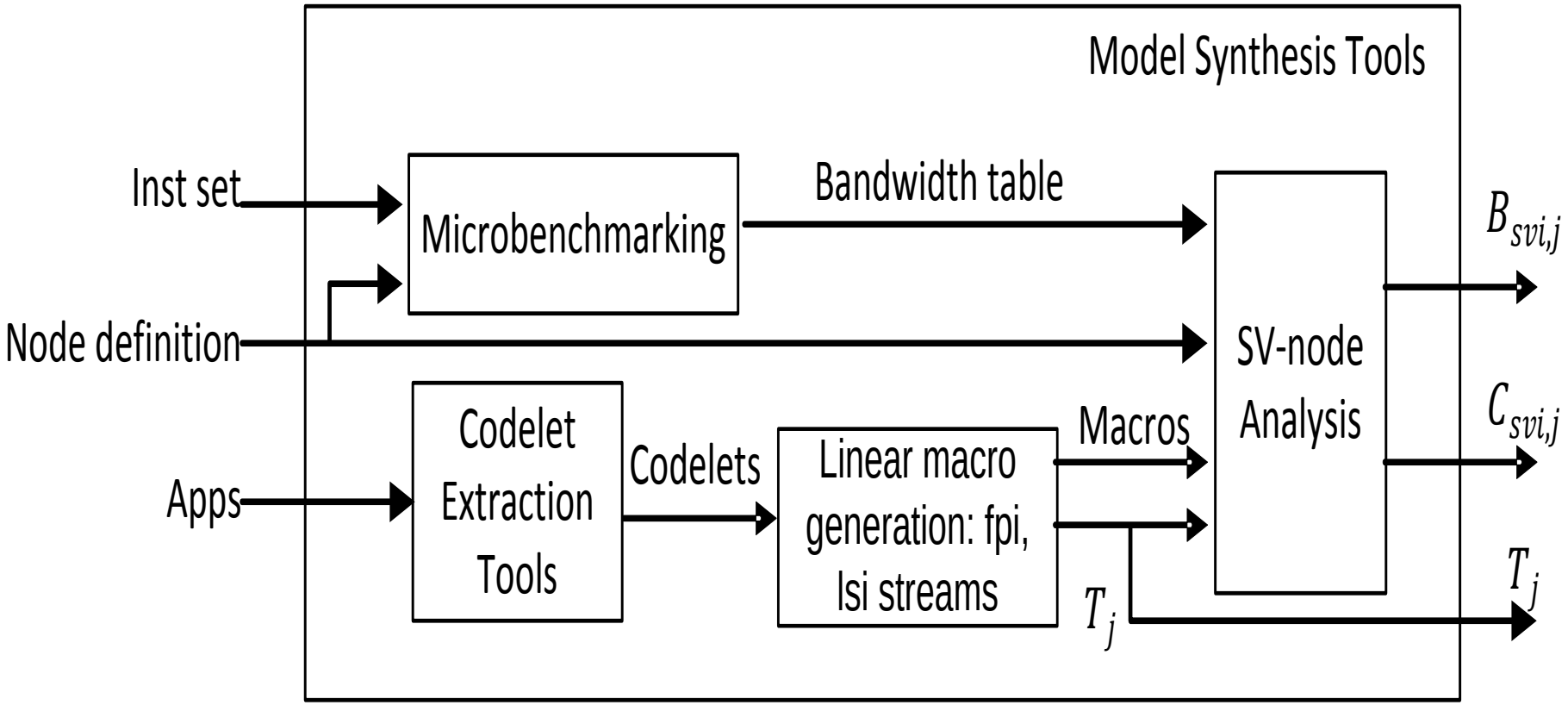
Codesign goals → Capture Complexity: may be nonlinear

D. Measurement: sources, types of data

- Simulator
  - RTL level: all details, phase=inst or more, very slow
  - Functional: less detail, faster
  - Numbers very arch-specific, constrains codesign variations
- Math model, e.g. queueing
  - Fast, but localized, and may be architecture constrained
- HW counters
  - Very fast, fixed meas. points, quirky (defs tricky, changing)
- **Single-valued *virtual nodes* (details following)**
  - Combine HW/SW at intuitively useful level
  - Flexible, allows various architectural mappings
  - Measure via binary instrumentation, nopping, ubenchmarks
  - Example: mem-mem vector ops: f(stride, length, …)

Input assumptions → output interpretation

D. Measurement: Decan SW + HW counters



Joint work with Intel Exascale Lab, Paris

Codelet → sv-node decomposition

Level	Properties
Original program → phases	Irregularities removed, e.g. -- alignment, aliasing, …
Codelet	Significant time, automatically isolatable, similar μ values, …
Macro	Mutually exclusive inst. seq., i.e. satisfy time linearity test
Single rate v-node	Similar phy. node execution and memory access, so constant execution rate

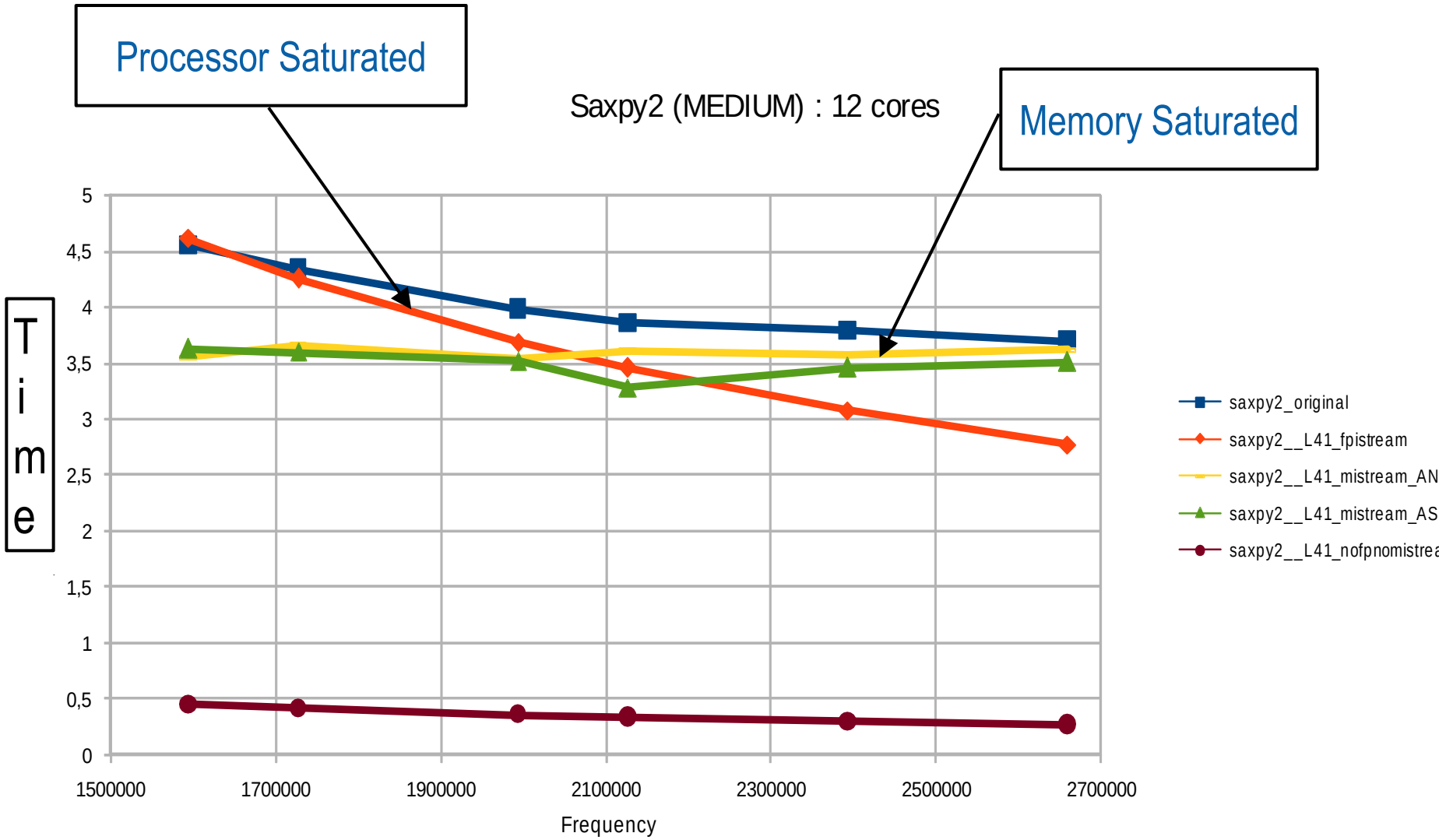
Tools list: next foil

SW Tools for sv-node modeling

- Microbenchmarking – node $B = \max_k \{C_{i,k}\}$
  - Generation tool
- Capacity analysis – <*node,phase*> *C*
  - Maqao: static analysis of assembly code
  - Decan: dynamic analysis of binary code
    - Replace selected instructions, run modified binaries
    - Nopping – change, kill, or replace op with nop
    - Destroys semantics, but gets accurate Capacity values
- Intel Exascale Lab – W. Jalby, Versailles

Tools for Application Characterization

Decan: Magma Codelet Behavior (2)



E. Cape Modeling Results

1. **Recommender**
   - Select best system from list of designs
2. **Simulation**
   - Explore design space
3. **Optimal codesign**
   - Best C/E for SW workload and HW constraints

Cape implemented by David Wong, Intel

E1. System Behavior – Quality Objectives

- *D*-Stability(unicore, *perf range*, *Data range*)
- *f*-Scalability(*freq*, *D*) [*D*-range vs. *f*]
- *p*-Speedup(*proc*, *D*) [how many cores/chip?]
- Energy, Energy efficiency, Power, … [many questions]
- Cost – initial, operating [how do we define?]
- Combinations of the above [how do we define?]
  - Depending on system type
  - Server, laptop, handheld, …

Goal: HW/SW codesign methodology that handles all of this, to maintain contracts with ISVs, OEMs, & end users

E.2 Capacity-based Recommender System

- When a user asks about purchasing a new system
  - Current websites give extensive lists, little insight
  - Reco tool recommends top choices
    - Perf ranking among user's options
  - Explains why chosen, based on current SW apps
    - v-nodes represent user-specific HW/SW combinations
- OEM customer-support program feature
  - Anonymously measuring 8M users continuously
  - Run "capacity model" periodically on user's system
- System model constrained
  - Apps include all processes running: 1 sec. samples
  - Increasing HW nodes selectable

Apps usage of system?: Users/OEMs don't understand

Simulation Example: 2 NR Codelets

Hqr13 Source:

```
SUBROUTINE codelet (n, m, i, a, res) ! m = 10, i = 1

integer n, j
real*8 a(m, n), s, res (1)

s = real (0)

do j = 1, n
      s = s + abs (a (j, i))
end do

res (1) = s

END SUBROUTINE codelet
```

Reduction

Svdcmp11 Source:

```
SUBROUTINE codelet (n, m, i, a, f) ! m = 20, i = 1

integer n, m, j, i
real*8 a(m, n), f

do j = 1, n
      a (i, j) = a (i, j) * f
end do

END SUBROUTINE codelet
```

Wrong
FTN index

- Which has good or bad: Stab[*D*]<2, Scal[*f,D*] S*p*[*D*]?

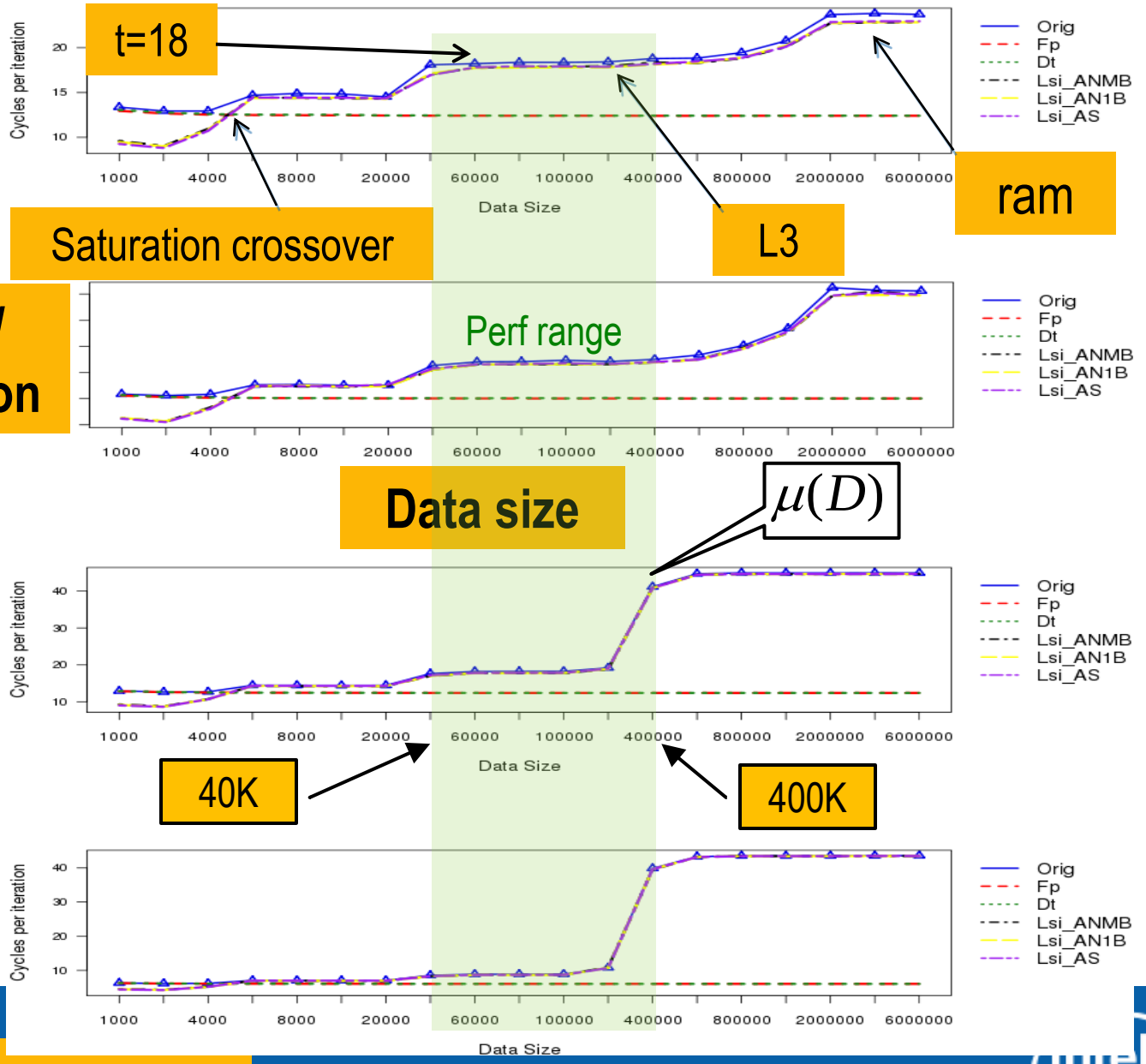
Codelet 1: Sp(4), Sf(2x) vs. *D*

1core/low freq

1c/hi f

4c/lo f

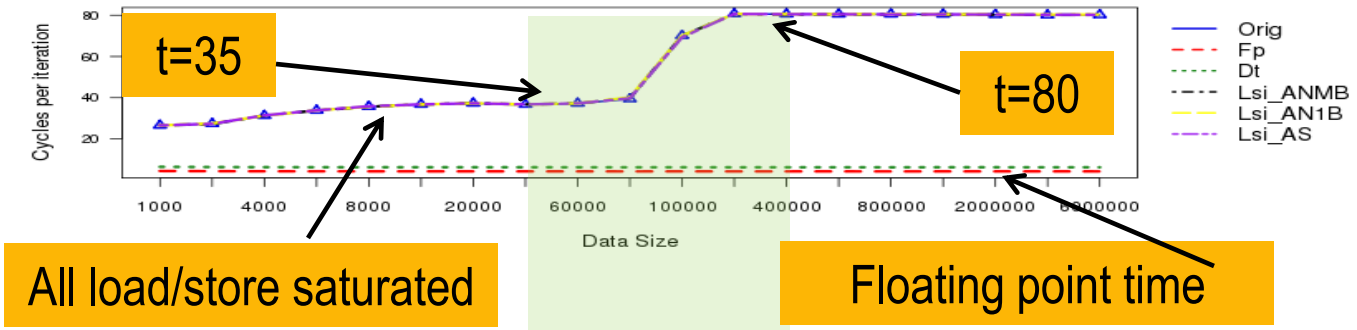
4core/high freq



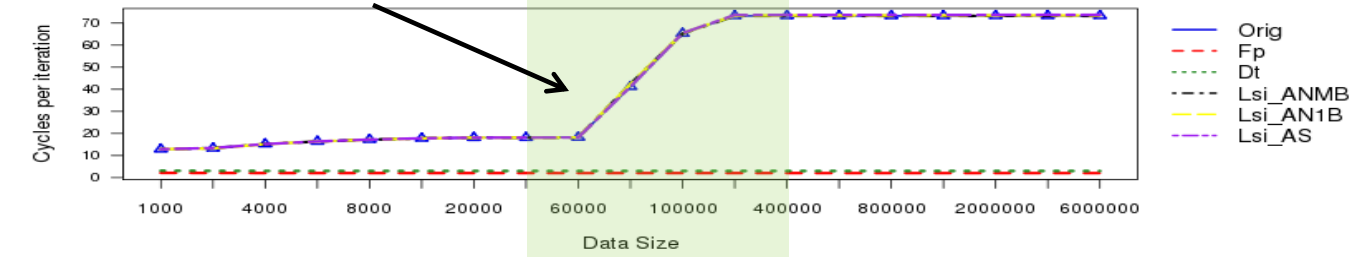
Hqr_13_dp_sse

Codelet 2: Svdcmp_11_dp_sse

1c/lf

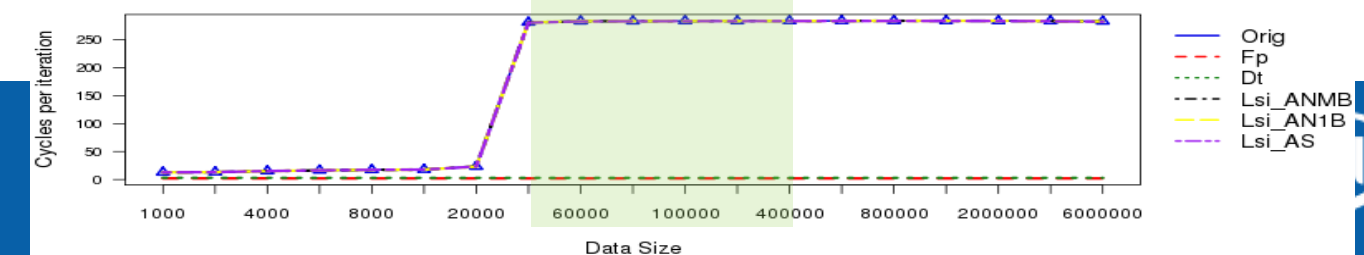


1c/hf



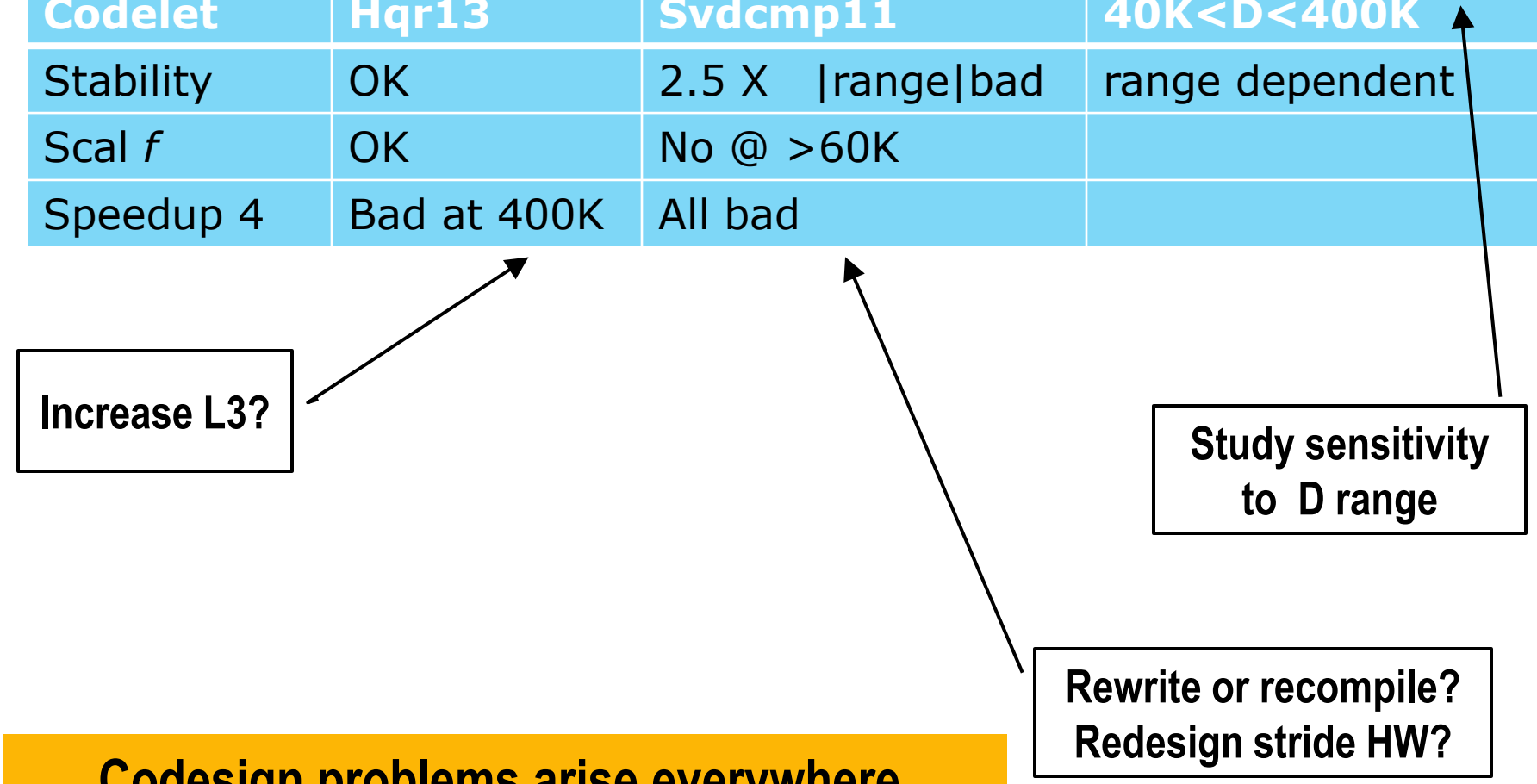
4c/lf

4c/hf



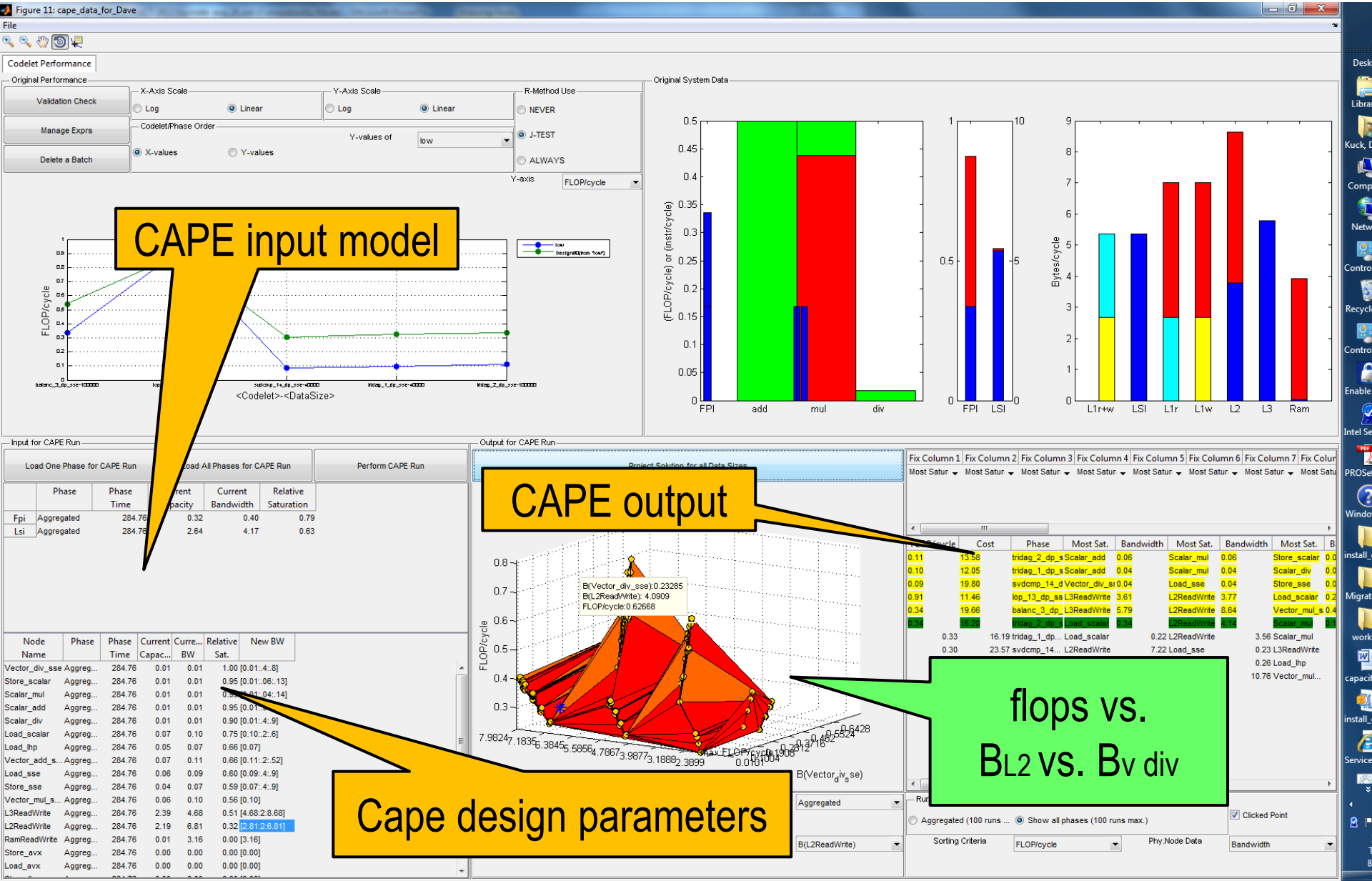
Quality Results

Codelet	Hqr13	Svdcmp11	40K<D<400K
Stability	OK	2.5 X |range|bad	range dependent
Scal *f*	OK	No @ >60K	
Speedup 4	Bad at 400K	All bad	



Codesign problems arise everywhere. Note that HW and/or SW changes are suggested by Q violations

5 phase (mixed lsi & fpi sat) – 35 node simulation



Example: 2X flops, .7 BW L2, 5X arith & L1

Problem

- 5 codelet workload: 3 proc bound, 2 mem bound
- Attempt to cut BW L2 while doubling perf

BW	Sc+	Sc*	Sc/	v/	L s	Lsse	S s	Ssse	L2 r/w	L3 r/w	Agg flops
Orig	.06	.06	.04	.04	.11	.44	.06	.44	10.2	6.0	.32
Design	.26	.19	1.2	.26	.34	2.3	.26	2.86	7.2	11.1	.63
Ratio	4x	3x	30x	6x	3x	5x	4x	6x	.7x	2x	2x

Preliminary output: node units not normalized

Cape Result (no optimization, 5 min manual search)

- L2 BW cut to .7 original (energy savings) [area vs. size]
- Main cost is 5X arithmetic speed, and register/ L1 access
- Optimization can exploit all such opportunities

This is a typical consequence of *fpi* vs. *lsi* interaction; syst perf vs. node BWs
Surface irregularities are hard to predict intuitively.

3. Cape optimal codesign inputs/outputs

- **Performance**
  - Minimal thresholds or step ahead: codesign process input
  - Bandwidth used units: input/output
- **Costs**
  - Initial cost = BW needs, operating cost = *E* etc.: input/output
  - Max limits
  - Variable as function of value to buyers of design
- **Load**
  - Defined using *node* $C_{i,j}$, μ, and saturations
  - Data sets → computation program paths: vary *phase weights*
  - Stability of design
    - *How sensitive are perf and cost to load-usage uncertainty?*

Whatever is not Input, tool chooses as Output

E. 3 example problems, Cape solutions

1. Min *cost*, max *perf* codesign problems
   a. Analytical models of critical breaks in codesign space
   b. Cape tool for *stable* codesign
2. From system set, choose max *perf* (or min *E/C*)
   a. Recommender system for OEM vendor website
3. Codesign energy efficient systems
   a. Offline phase analysis predicts future
      → online (*f,V*) control governor [$B^{waste} \rightarrow 0$]
   b. min *E* or min *E/C* solutions

Treat measured μ ratios as constants

Deals with complexity that humans cannot

D.1. Solving *cost/perf* codesign problems (3node X 2phase) example

ata Arrangement

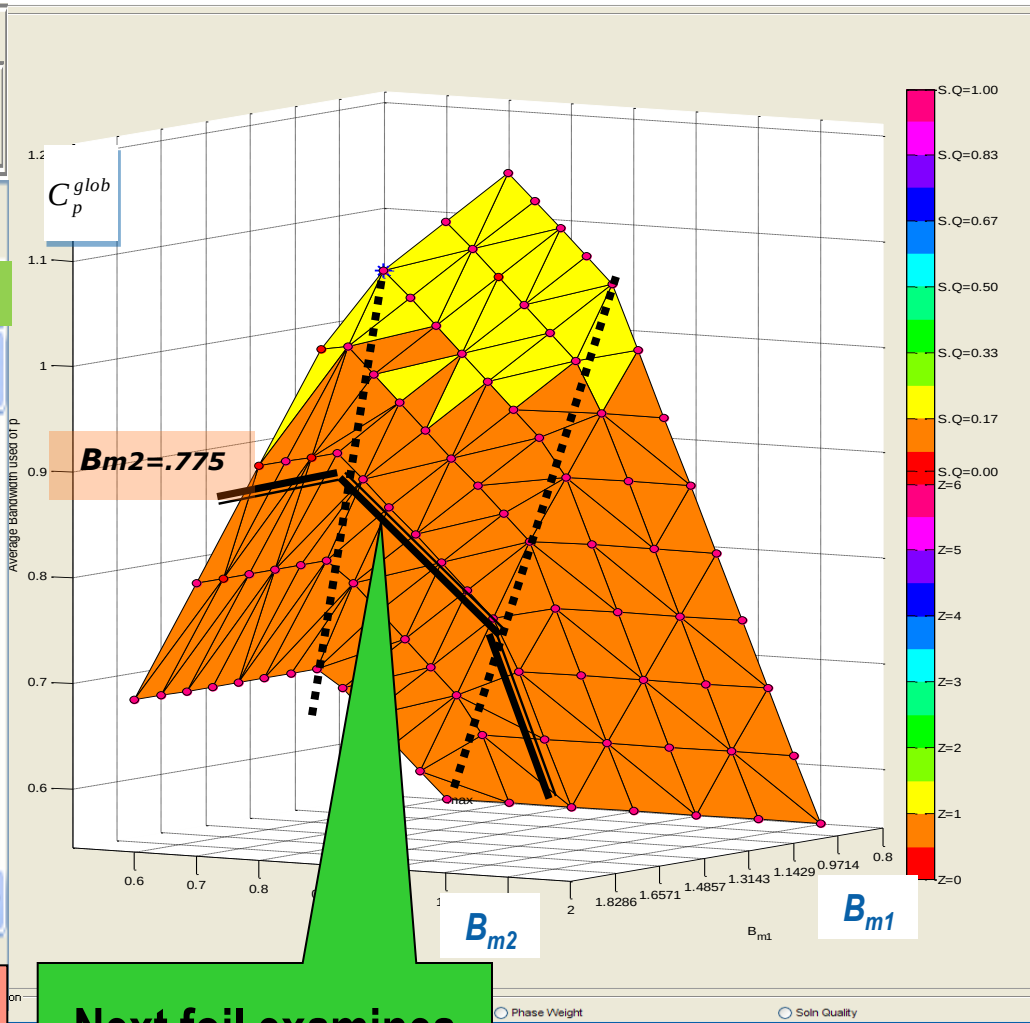
ata Group by

Data Sorting/Filtering

Sort Ascendingly Sort Descendingly

Select Min Select Max UnSelect M...

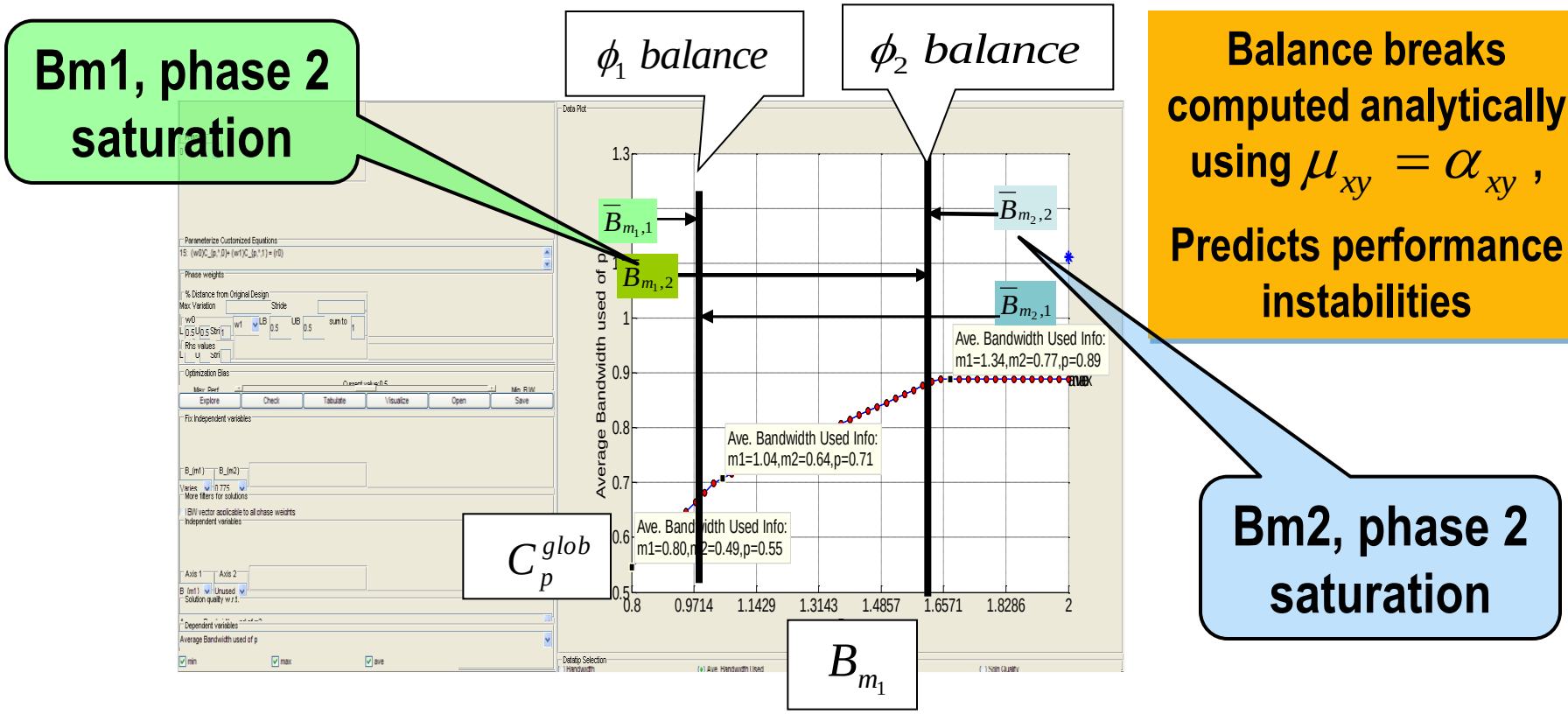
Cost	maxPerf(p)	Perf range(p)	m1	m2	p
4.1	1.1110	0	**1.7**	1.2000	**1.2**
4.2000	1.1110	0	1.7000	1.2000	1.3000
4.2000	1.1110	0	1.8000	1.2000	1.2000
4.2000	1.1110	0	1.9000	1.1000	1.2000
4.3000	1.1110	0	1.7000	1.2000	1.4000
4.3000	1.1110	0	1.8000	1.2000	1.3000
4.3000	1.1110	0	1.9000	1.1000	1.3000
4.3000	1.1110	0	1.9000	1.2000	1.2000
4.3000	1.1110	0	2	**1.0**	1.3000
4.3000	1.1110	0	2	1.1000	1.2000



Given perf = 1.11, cost = 4.3
Each *B* varies by 20%

Next foil examines this *Bm2* =.775 cut

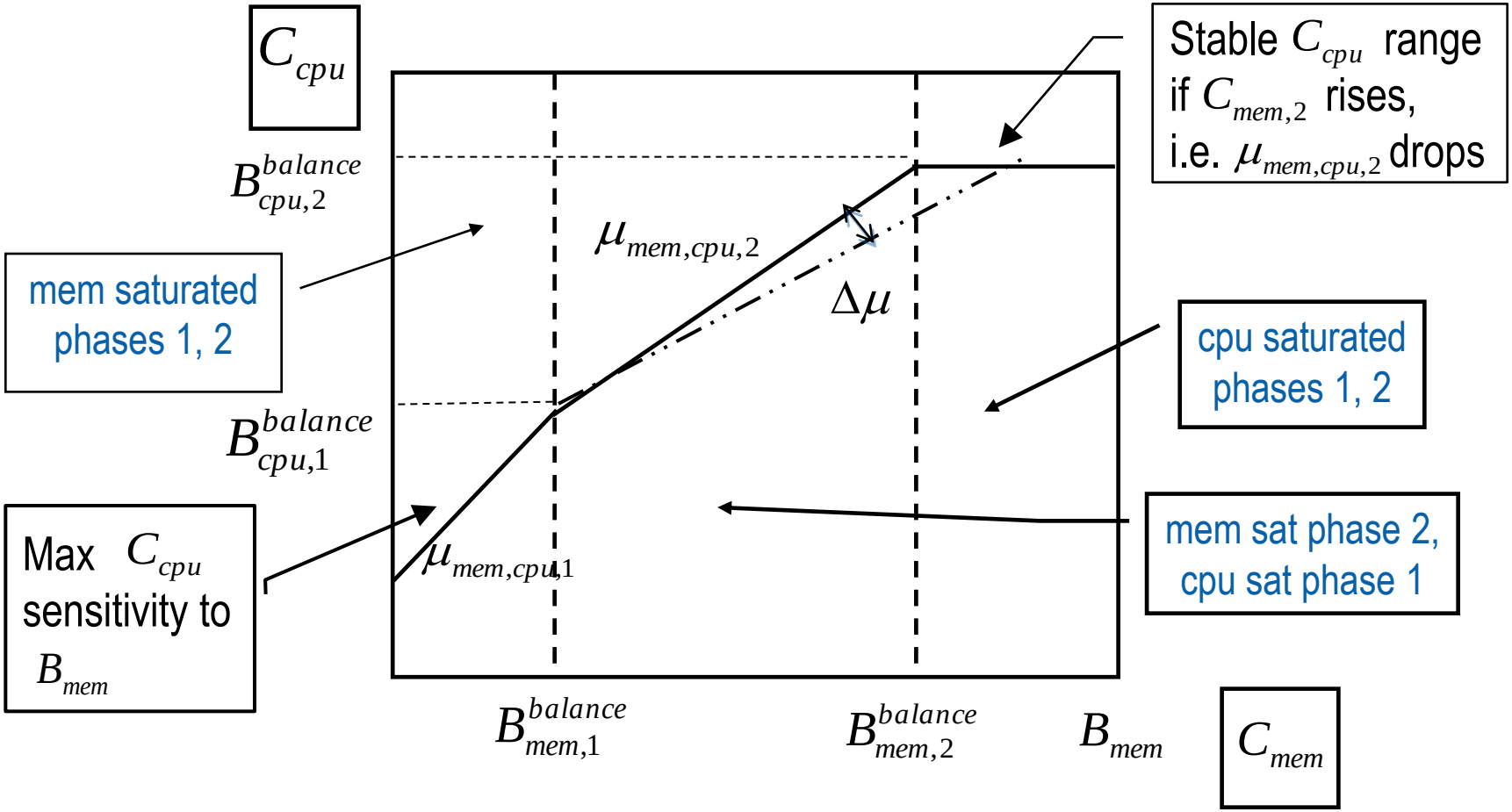
D1. cont. Sensitivity of Performance to the System



Processor perf vs. B_{m1}, showing 3 perf regions; B_{m2}=.775

Vizualize on surface, understand analytically

Mem BW vs. Perf. Stability?



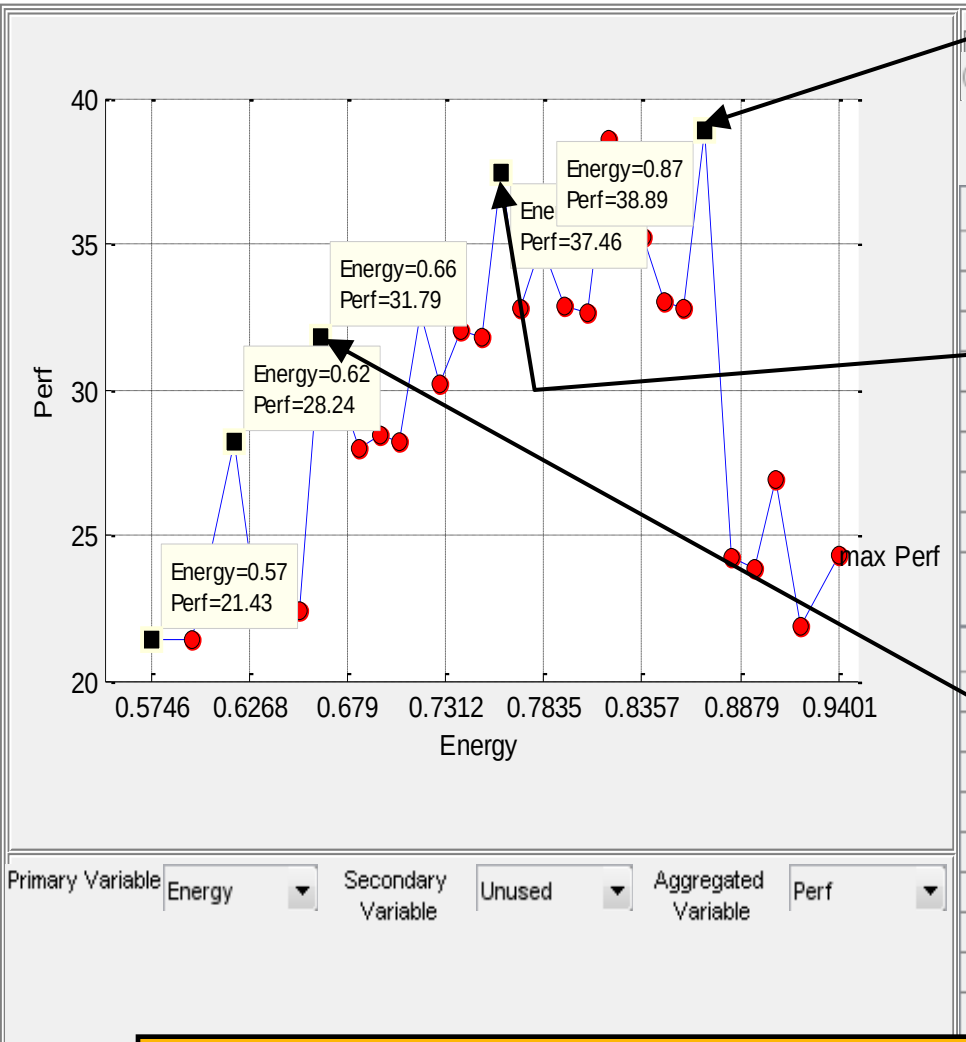
$$B^{balance}_{mem,i} = \mu_{cpu,mem,i} B_{cpu} = B_{cpu} / \mu_{mem,cpu,i}$$

3-node, 2-phase balance points

D.3. Power and Energy Objectives

- Design-in low power model: W_i^{idle} , $W_i^{\max}$, γ_i
  - *C/E or C/W proportional computing: use W an*d E efficiency
    - Market needs depend relative loading of systems
- Keep instantaneous power < thermal limit
- Run-time energy control (*f,V*) scaling, DVFS
  - P and C states: *C* for various idle *W* levels, P for (*f,V*) levels
    - Energy and time consumed making transitions
  - Race-to-Idle: only useful if *W* model is sufficiently poor
    - Conditions easy to state using the model
  - Multiprogramming complications if all cores scale together
- Preprocess apps for phase-level (*f,V*) self-scaling
  - OS scheduling interactions, depending on (*f,V*) resolution

3.a Perf vs. Energy for (2x3) model 3 W-states



max perf C= 38.9
E =.87

min E/C
perf = 37.5 (-3.6%)
energy = .77 (-11.7)

perf = 31.8 (-18%)
Energy = .66 (-23%)

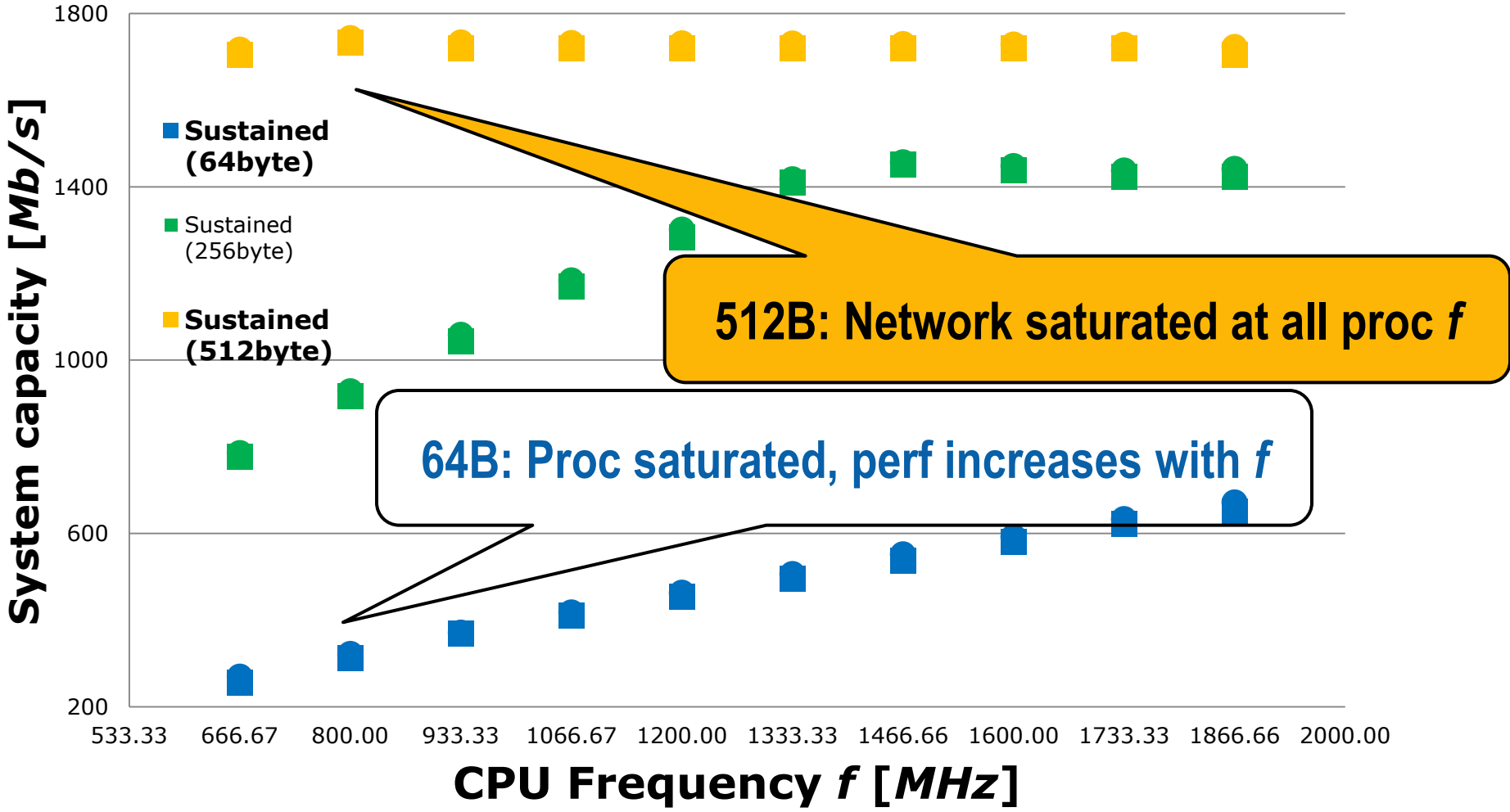
Solution Row Display
Show All Rows
Show Highlighted Rows
Primary Sorting Criteria: Perf
Num. Displayed for Primary Sorting Criteria: 1
Secondary Sorting Criteria: Power

	Perf	Power	Energy	Energy/T...	Cost	BW(cpu,1)	BW(gpu,1
1	38.9000	33.8659	0.8708	2.9272e-04	3.7333e+03	1.8667e+03	1.8667e+0
2	38.6000	31.6588	0.8195	2.7738e-04	3.7333e+03	1.8667e+03	1.8667e+0
3	37.7000	30.9290	0.8202	2.8438e-04	3.7333e+03	800	1.8667e+0
4	37.5000	28.7995	0.7680	2.6831e-04	3.7333e+03	800	1.8667e+0
5	35.2000	29.4366	0.8357	2.8107e-04	3.7333e+03	666.6667	1.8667e+0
6	35	27.4581	0.7845	2.6298e-04	3.7333e+03	666.6667	1.8667e+0
7	33	28.1072	0.8507	2.8661e-04	3.7333e+03	666.6667	1.8667e+0
8	32.8000	25.2754	0.8303	3.0667e-04	3.7333e+03	666.6667	1.8667e+0
9	32.8000	25.3012	0.7711	3.0719e-04	3.7333e+03	666.6667	120

	Ph. Time 1	Ph. Time 2	Ph. Time 3	BWU(cpu...	BWU(gpu...	BWU(cpu...	BWU(g
1	0.0086	0.0086	0.0086	875.00	1866.67	1225.00	186
2	0.0086	0.0086	0.0088	875.00	1866.67	1225.00	186
3	0.0086	0.0086	0.0086	875.00	1706.67	1225.00	186
4	0.0094	0.0086	0.0088	800.00	1706.67	1225.00	186
5	0.0113	0.0086	0.0086	666.67	1422.22	1225.00	186
6	0.0113	0.0086	0.0088	666.67	1422.22	1225.00	186
7	0.0086	0.0131	0.0086	875.00	1866.67	800.00	121
8	0.0086	0.0133	0.0086	875.00	1866.67	787.50	120
9	0.0133	0.0086	0.0086	562.50	1200.00	1225.00	186

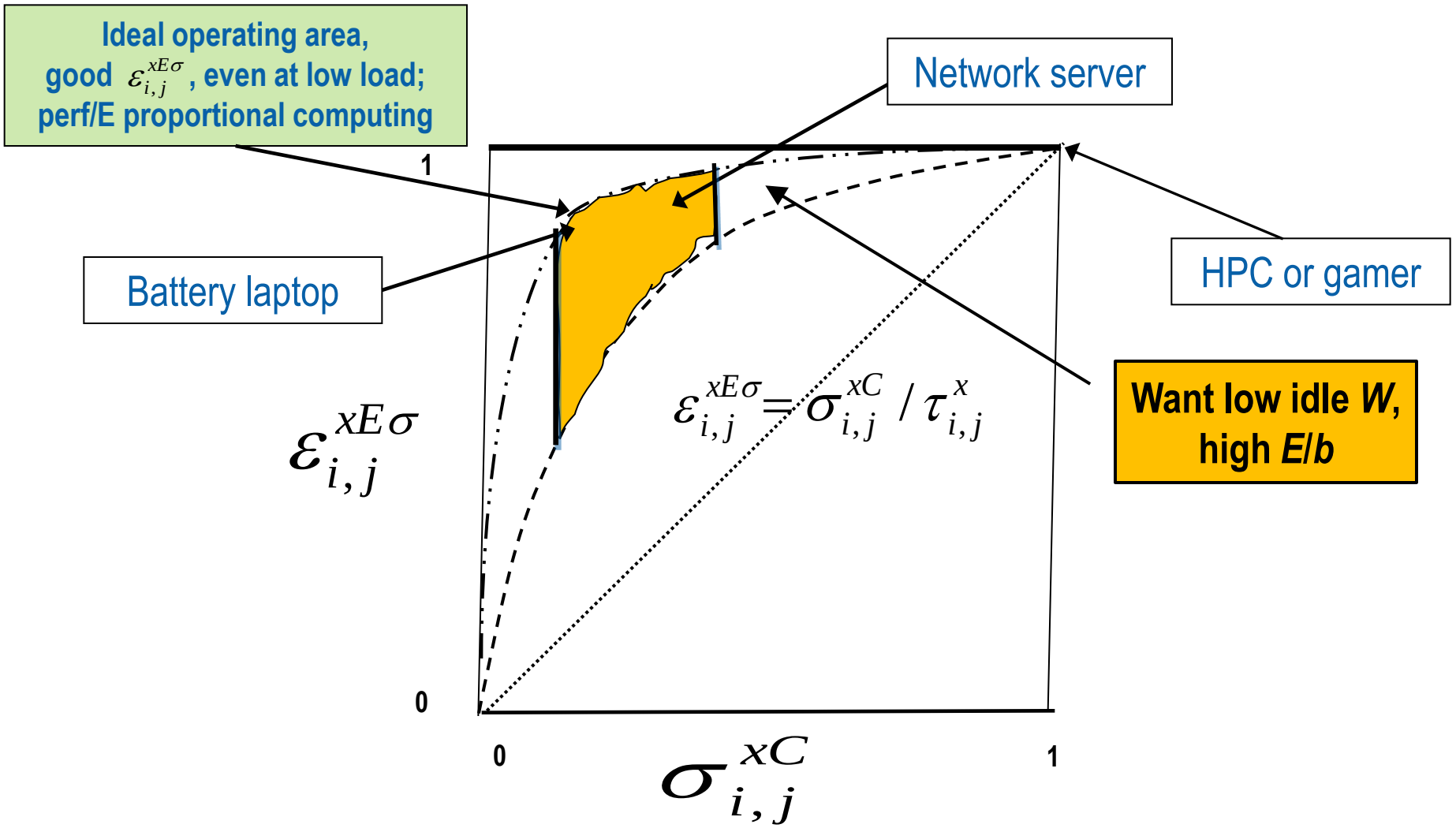
Codesign Pareto front (black dots)→ Design choices

3.b Network perf results



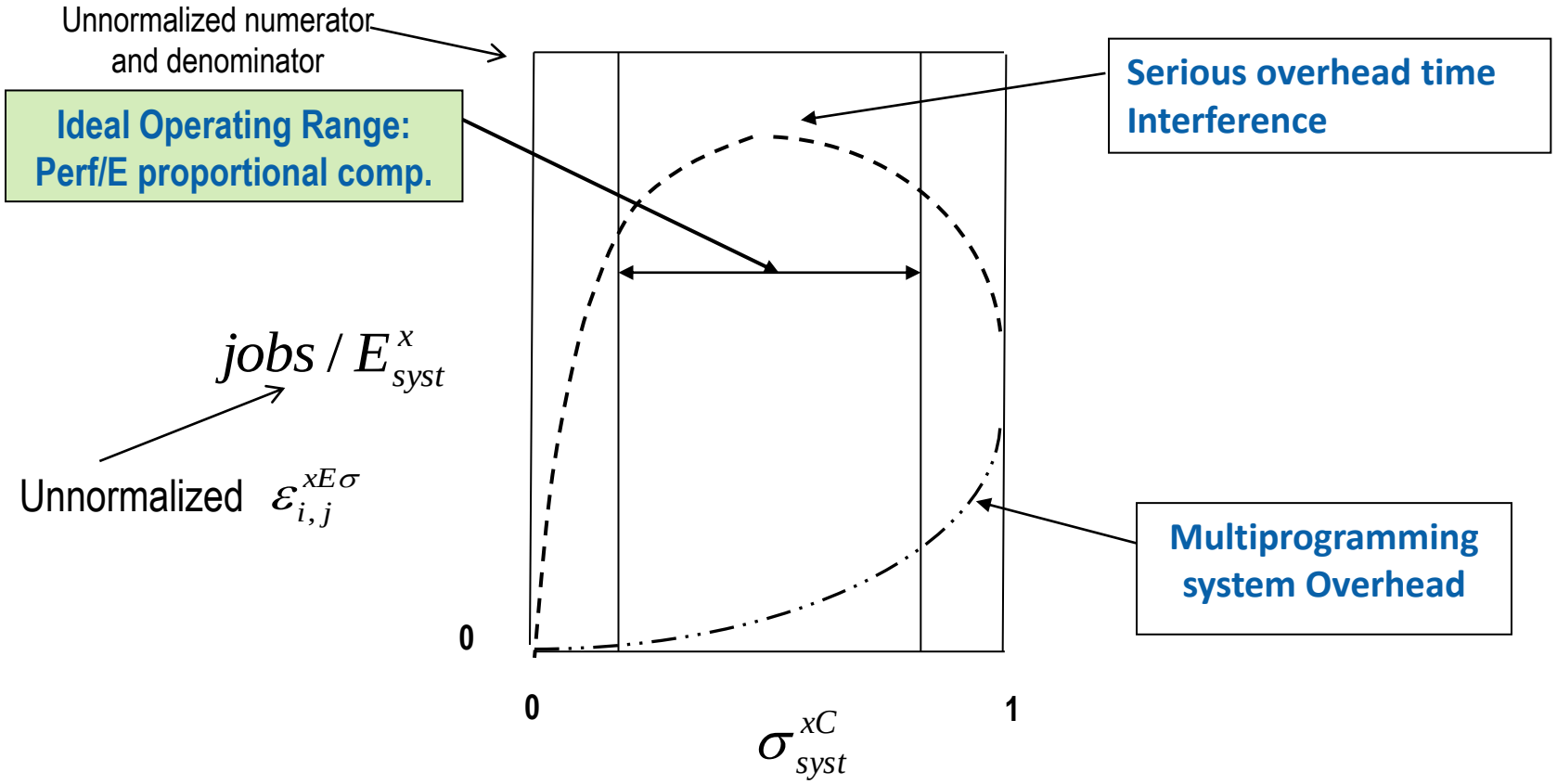
Packet size → system behavior including *E(f,V)*

Energy efficiency vs. Capacity Rel. Saturation



$$\text{energy efficiency} = \varepsilon_{i,j}^{xE\sigma} = \frac{\sigma_{i,j}^{xC}}{\sigma_{i,j}^{xyE}} = \frac{C_{i,j}^{x} W_{i,j}^{yE\max} t_{i,j}^{yE\max}}{B_{i,j}^{x} W_{i,j}^{x} t_{i,j}^{x}} = \frac{E_{i,j}^{\max}}{E_{i,j}}$$

Multiprogrammed jobs/energy vs. Capacity relative saturation



"Server consolidation" problem

Key benefits of capacity-based codesign

1. Top-down codesign of optimal systems

- Include system-wide interactions
- Mixed fidelity saves modeling effort and simulation time

2. Simultaneous use of all "known" load/BW info

- Overcome human-limiting *complexity* via automatic process
- Capture parameter *uncertainties* via sensitivity analysis

3. Design focused-system families

- Cluster usages partition market by HW needs
- Specialized system-per-market always beats general systems
- System-family codesign softens combinatorial explosion

Fast optimization (coherent data) → Codesign results

References

- [JWKA11] William Jalby, et al, *Measuring Computer Performance*, in HPC, Springer, 2011.
- [KFMJ10] Souad Koliai, Fursin, Mosely, Jalby, "DECAN: Decremantal Analysis via Binary Patching, Draft Apr. 2010.
- [Kuck74] D. J. Kuck, "Computer System Capacity Fundamentals," National Bureau of Standards, Technical Note 851, Oct. 1974.
- [KuKu76] D.Kuck, B. Kumar, A system model for computer performance evaluation, SIGMETRICS '76: Proc 1976 ACM SIGMETRICS
- [Kuck78] D.J. Kuck, The Structure of Computers and Computations, Vol. I, John Wiley & Sons, Inc., 1978.
- [Kuck11] DJ Kuck, *Computational Capacity-based Codesign of Computer Systems,* in HPC, Springer, 2011.
- [LiLS08] Lixia Liu et al, "Analyzing memory access intensity in parallel programs on multicore", *Proc. 22nd ACM ICS,*pp. 359-367, June 2008.
- [NPJW13] J. Noudohouenou, et al, Simsys: *A Performance Simulation Framework*, RAPIDO13, 2013, Berlin.