UNIVERSITAT JAUME·I

Energy-Aware Linear Algebra

Enrique S. Quintana-Ortí

Concurrency and energy efficiency

UNIVERSITAT JAUME·I

- Green500 vs Top500 (June 2013)

Rank Top/Green	Site	Technology	MFLOPS/W
1/32	Tianhe-2 - National University of Defense Technology	Intel Xeon E5 + Intel Xeon Phi	1.901
467/1	Eurora - CINECA	Intel Xeon E5 + NVIDIA K20	3.208

Concurrency and energy efficiency

UNIVERSITAT JAUME·I

- Green500 vs Top500 (June 2013)

Rank Top/Green	Site	Technology	MFLOPS/W	MW to EXAFLOPS?
1/32	Tianhe-2 - National University of Defense Technology	Intel Xeon E5 + Intel Xeon Phi	1.901	408
467/1	Eurora - CINECA	Intel Xeon E5 + NVIDIA K20	3.208	312

Most powerful reactor under construction in France
Flamanville (EDF, 2017 for US $9 billion):
1,630 MWe

Concurrency and energy efficiency

UNIVERSITAT JAUME·I

- Green500 vs Top500 (November 2012)

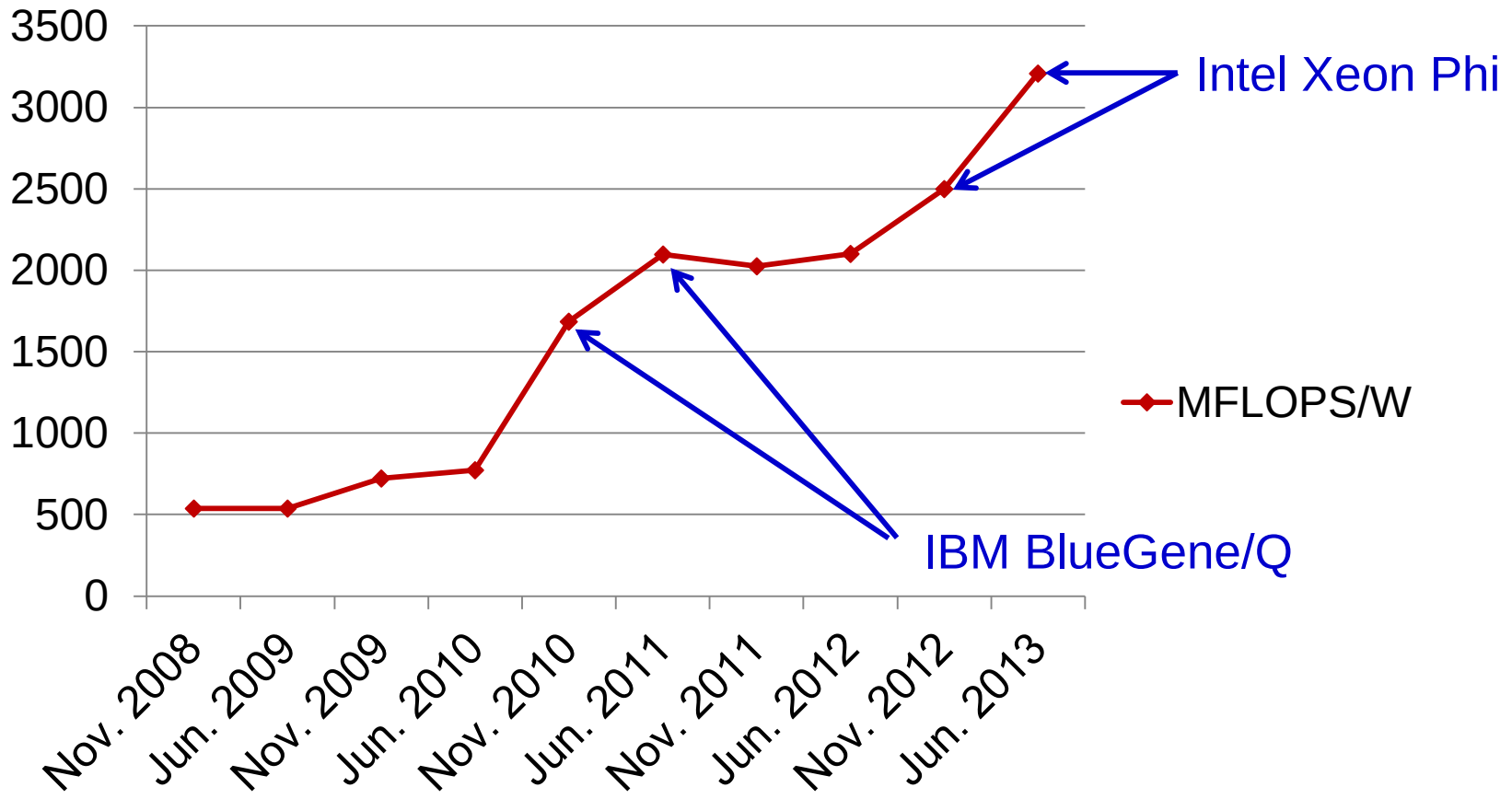
Rank Top/Green	Site	Technology	MFLOPS/W	MW to EXAFLOPS?
1/32	Tianhe- Univers Technology		1.901	408
467/1	Eurora - CINECA	Intel Xeon E5 + NVIDIA K20	3.208	312

1 MW ≈ $1 Million/year!

Most powerful reactor under construction in France
Flamanville (EDF, 2017 for US $9 billion):
1,630 MWe

Concurrency and energy efficiency

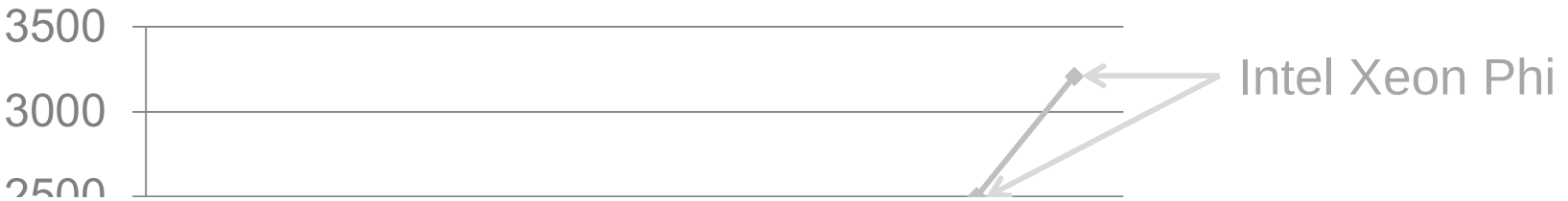
- System ranked #1 in Green500



Concurrency and energy efficiency

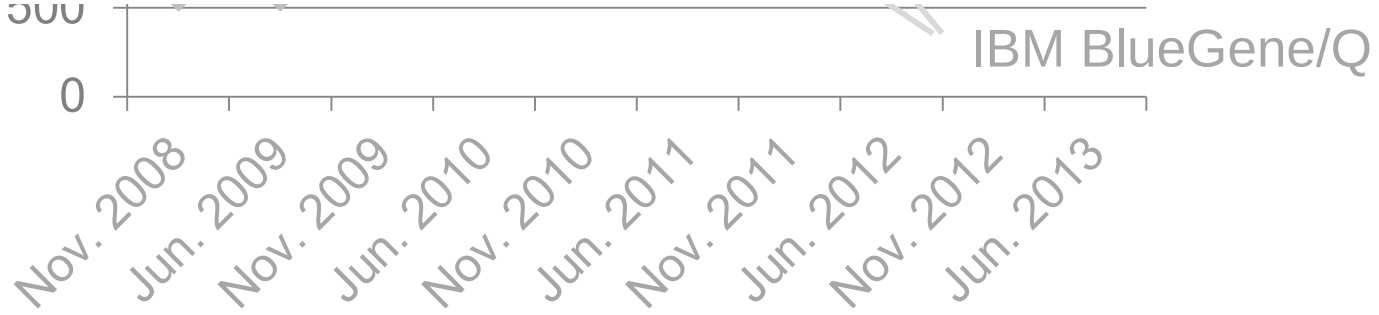
UNIVERSITAT JAUME·I

- System ranked #1 in Green500



Goal: 20MW for 1 EXAFLOP by 2020

Maintaining the improvement rate of last five years (x5) → 40 MW by 2020!!!



Concurrency and energy efficiency

UNIVERSITAT JAUME·I

- Reduce energy consumption!
  - Costs over lifetime of an HPC facility often exceed acquisition costs
  - Carbon dioxide is a hazard for health and environment
  - Heat reduces hardware reliability
- Personal view
  - Hardware features some power-saving mechanisms
  - Scientific apps. are in general energy-oblivious

Experimental setup

UNIVERSITAT JAUME·I

AMD

- 2 AMD Opteron 6128, 48GB
- DVFS per core

P-state	VCC_i	f_i
P_0	1.23	2.00
P_1	1.17	1.50
P_2	1.12	1.20
P_3	1.09	1.00
P_4	1.06	0.80

Intel

- 2 Intel Xeon E5504, 32GB
- DVFS per socket

P-state	VCC_i	f_i
P_0	1.04	2.00
P_1	0.98	1.73
P_2	0.95	1.60
P_3	1.01	1.87

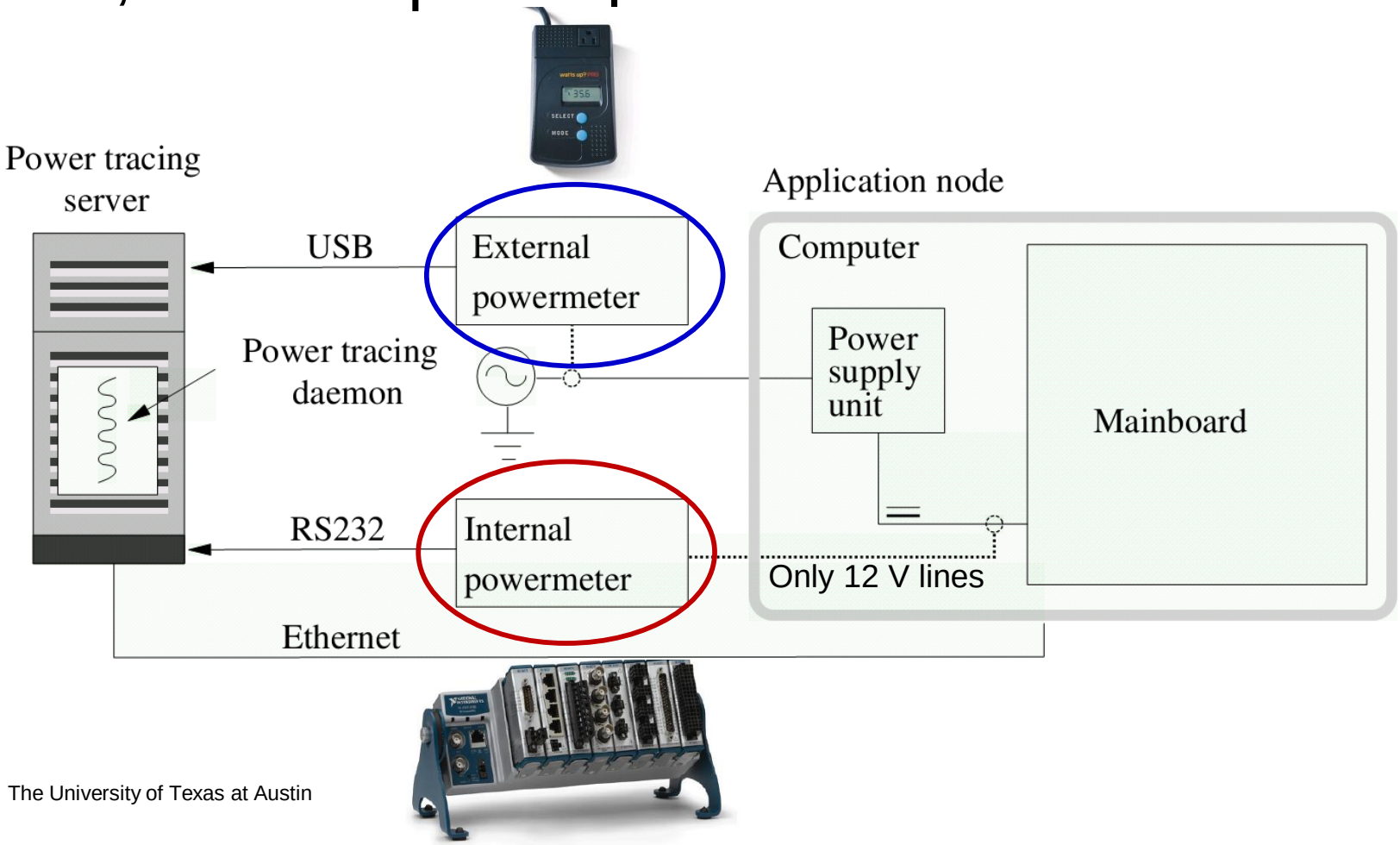
C-states:

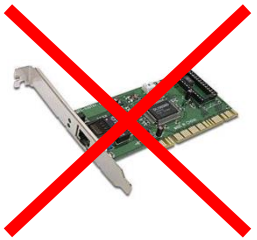
C0: normal operation mode

C1, C1E: disable core components (L1/L2 caches), clock signal, mem. controller,… increases energy savings at the expense of recovery time

Experimental setup

- National Instruments NI9205+NIcDAQ-9178
- 1,000 Samples/s per channel





Outline

UNIVERSITAT JAUME·I

- Modeling power
- Saving power in task-parallel applications
  - ILUPACK for multicore processors
  - CG for hybrid CPU-GPU platforms
- Conclusions

Outline

- Modeling power
- Saving power in task-parallel applications
  - ILUPACK for multicore processors
  - CG for hybrid CPU-GPU platforms
- Conclusions

Modeling Power

$$P = P^{(S)Y(stem)} + P^{C(PU)} = P^{Y} + P^{S(tatic)} + P^{D(ynamic)}$$

- P^C is the power dissipated by CPU (socket): $P^S + P^D$
- P^S is the static power
- P^D is dynamic power
- P^Y is the power of remaining components (e.g., RAM)

Considerations:

- P^Y and P^S are constants (though P^S grows with temperature)
- Hot system

Modeling Power

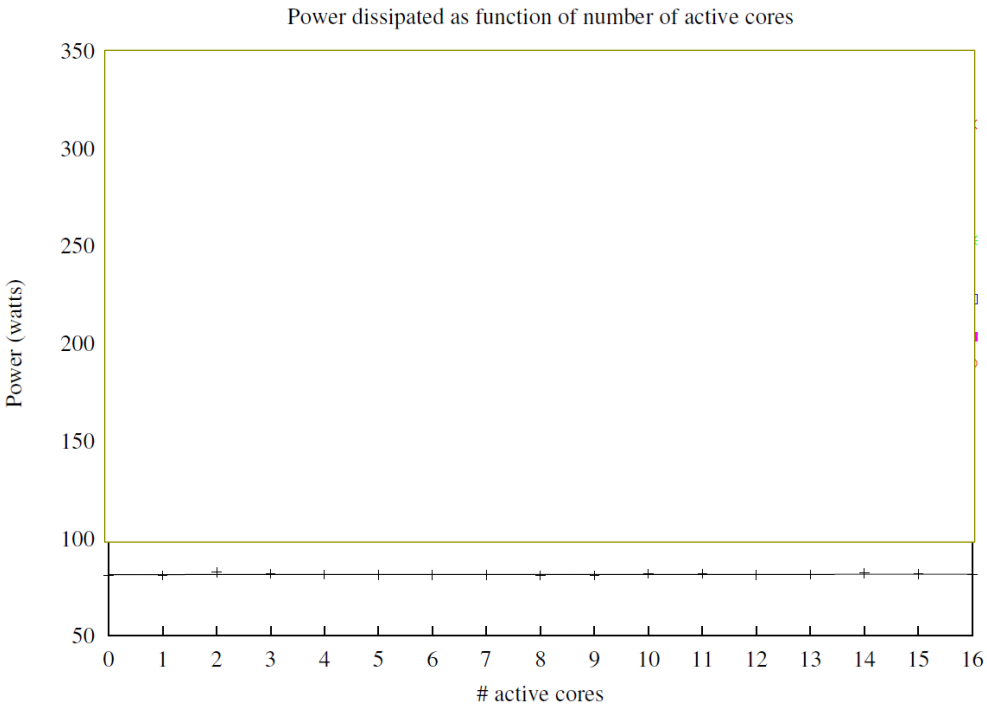
- System power:

$$P = P^Y + P^S + P^D$$

Estimated as *idle* power

Due to off-chip components:

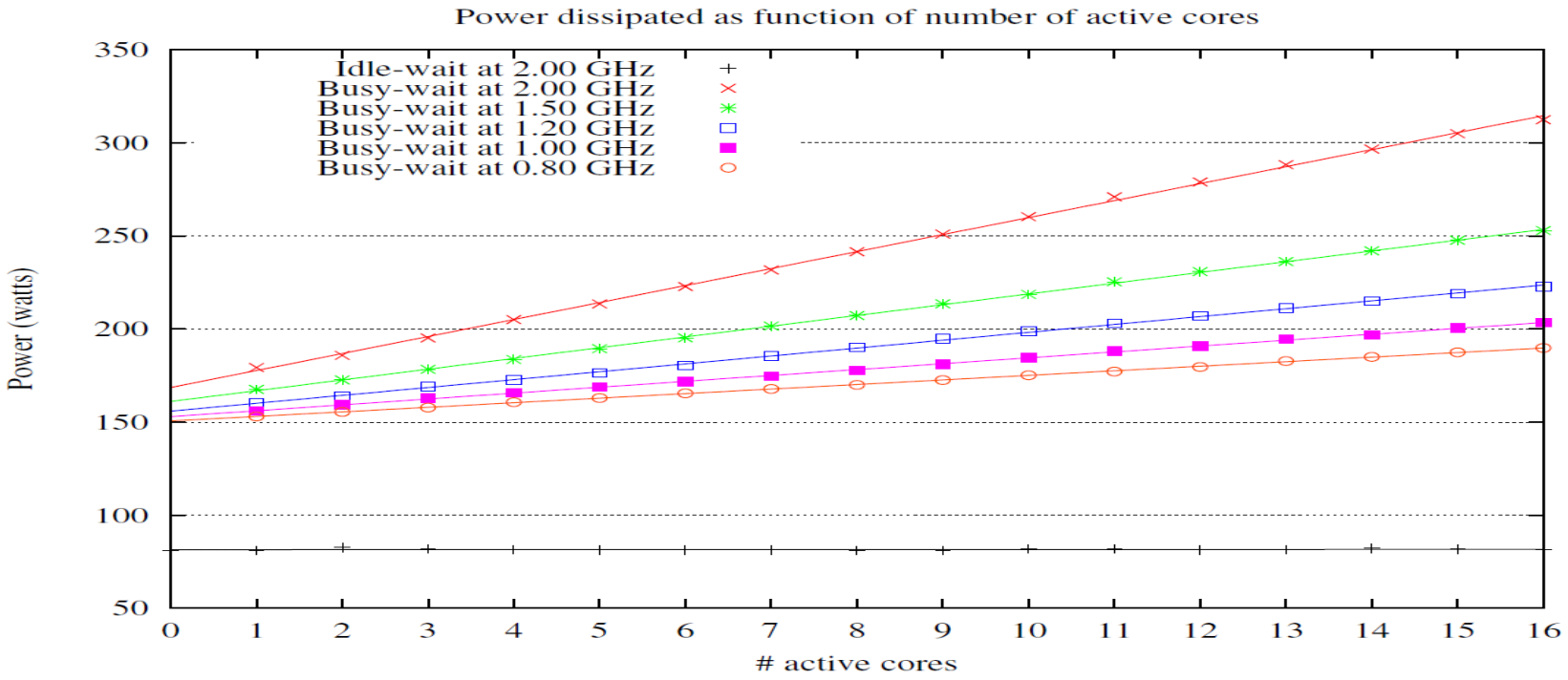
e.g., RAM (only mainboard)



$$P^Y \approx P^I = 80.15\ \text{W}$$

Modeling Power

- Static power: $P = P^Y + P^S + P^D$

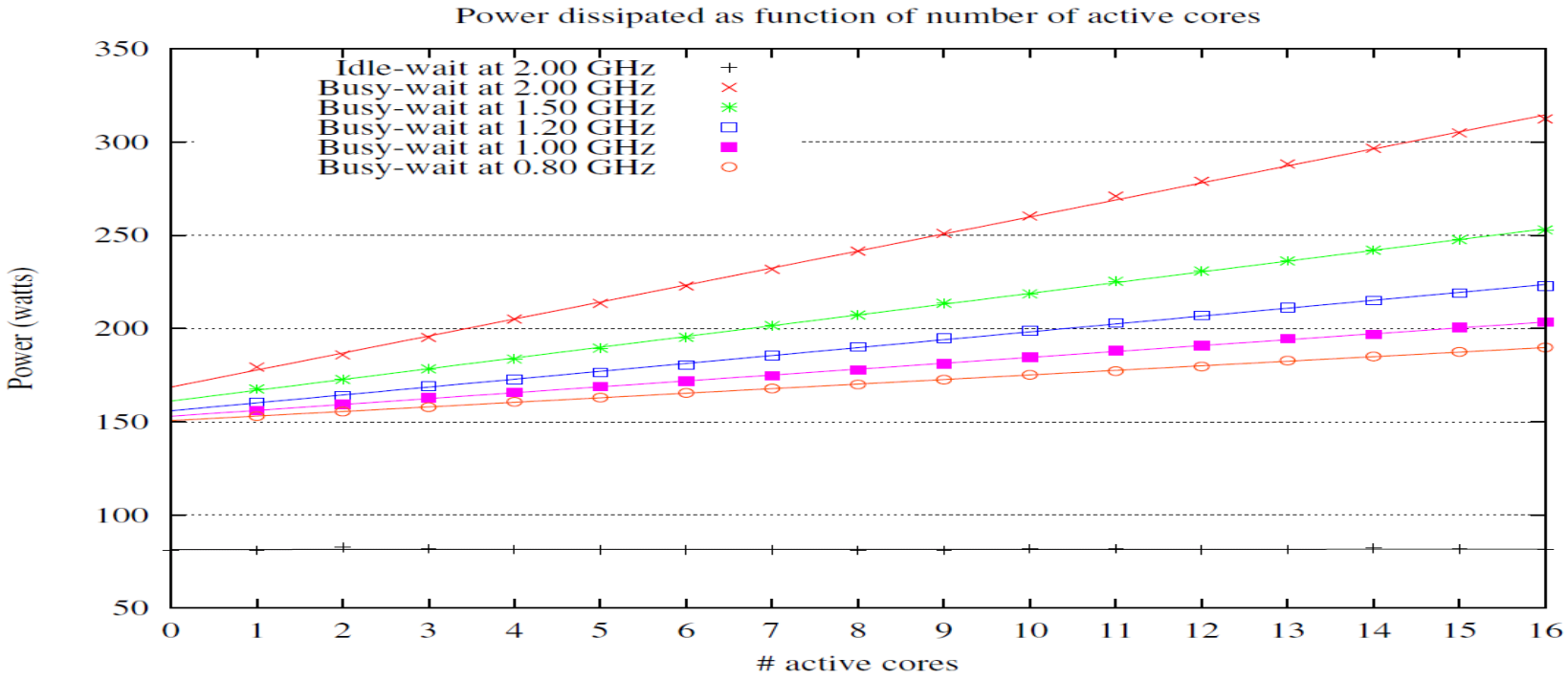


$$P^T{}_0(c) = a_0 + b_0 \cdot c = 168.59 + 9.12 \cdot c \text{ W}$$

$$P^S{}_0 \approx a_0 - P^Y = 88.44 \text{ W}$$

Modeling Power

- Dynamic power: $P = P^Y + P^S + P^D$



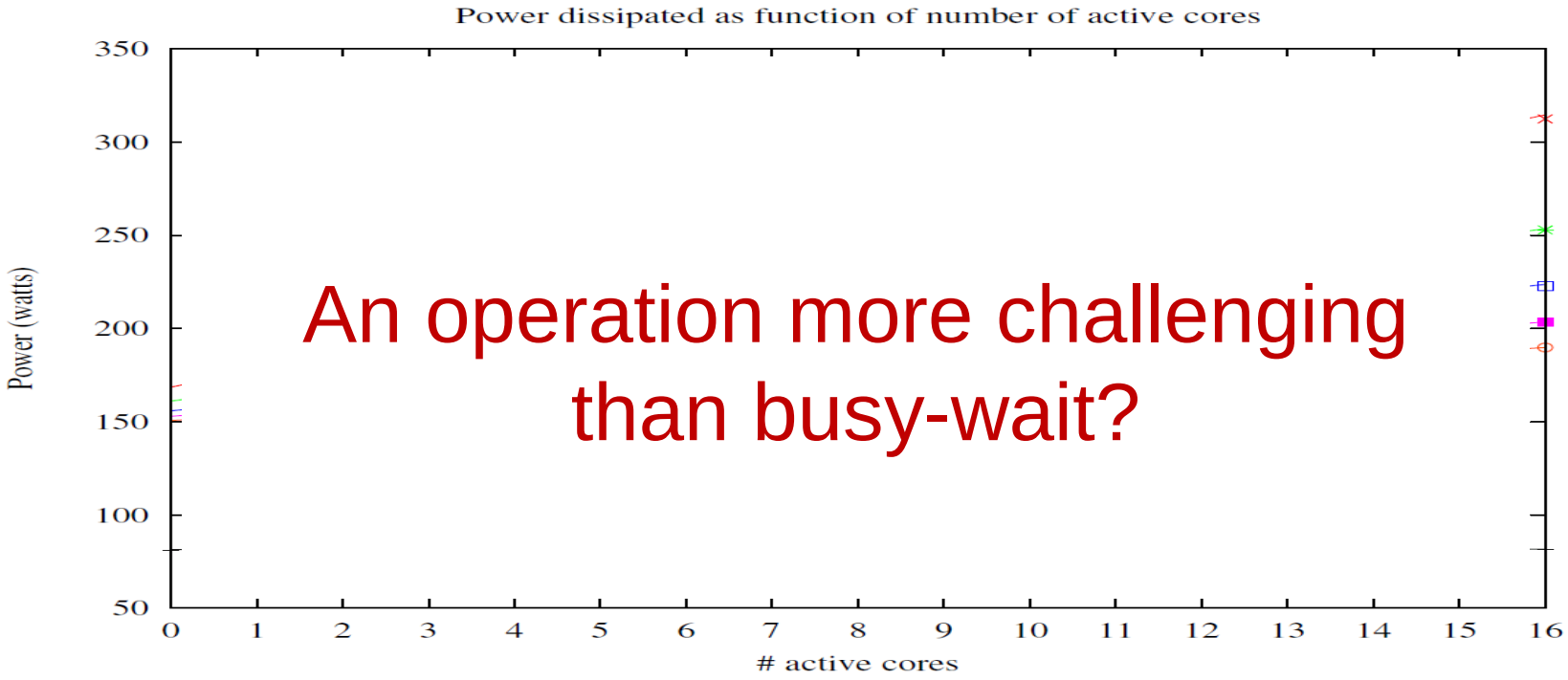
$$P^T{}_0(c) = a_0 + b_0\,c = 168.59 + 9.12 \cdot c \text{ W}$$

Busy-wait: $P^D{}_0 \approx b_0\,c = 9.12 \cdot c$ W

Modeling Power

- Dynamic power:

$$P = P^Y + P^S + P^D$$



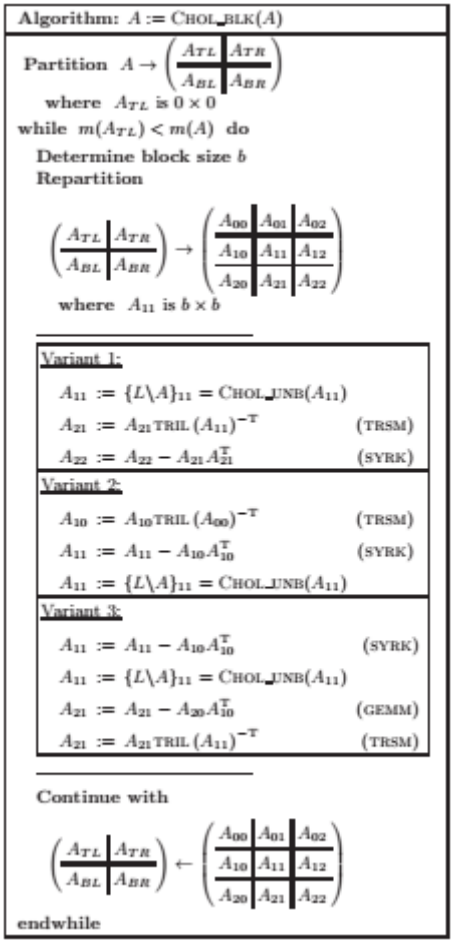
An operation more challenging than busy-wait?

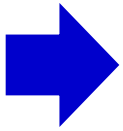
$$P^T{}_0(c) = a_0 + b_0 c = 168.59 + 9.12 \cdot c \text{ W}$$

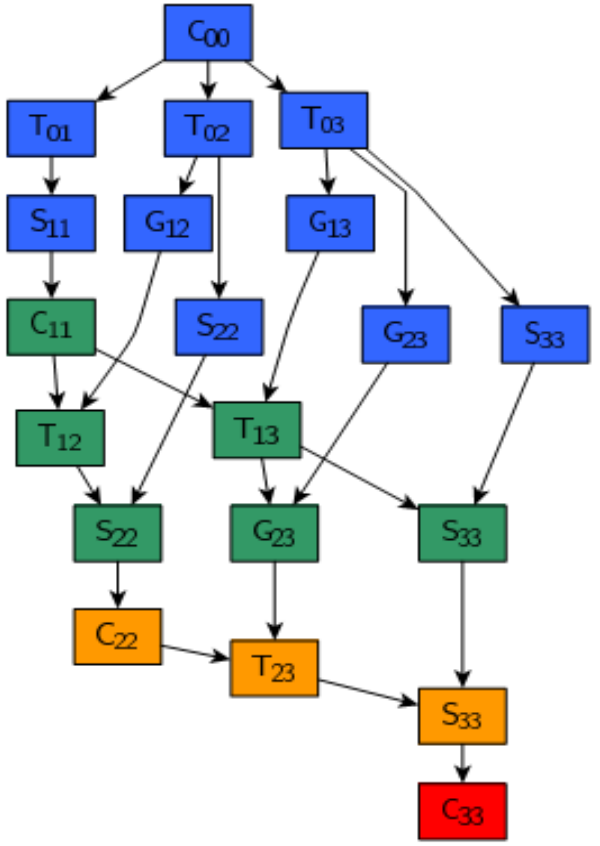
Busy-wait: $P^D{}_0 \approx b_0 c = 9.12 \cdot c$ W

Modeling Power

- Task-parallel DLA on multicore and CPU-GPU

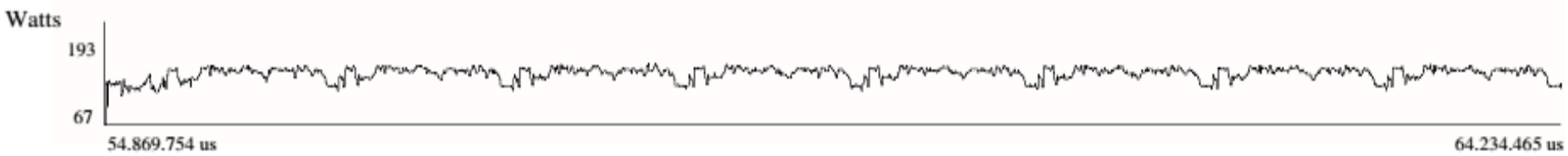






Modeling Power

- Task-parallel DLA on multicore and CPU-GPU



$$P_{\text{Chol}}(t) = P^U + P^S + P^D_{\text{Chol}}(t)$$
$$= P^U + P^S + \sum_{i=1}^{r} \sum_{j=1}^{c} P^D_i N_{i,j}(t)$$

	Block size, b			
Task	128	192	256	512
P^D_{P} (`dpotrf`)	10.26	10.35	10.45	11.28
P^D_{T} (`dtrsm`)	10.12	10.31	10.32	10.80
P^D_{S} (`dsyrk`)	11.22	11.47	11.67	12.60
P^D_{G} (`dgemm`)	11.98	12.54	12.72	13.30
P^D_{B} (`busy`)	7.62	7.62	7.62	7.62

- Use average Power
- Depends also on #active sockets!

Modeling Power

UNIVERSITAT JAUME·I

- Task-parallel DLA on multicore and CPU-GPU
  - Accommodate to memory contention

$$P_i^D \longrightarrow P_{\{i:j\}}^D = \delta_{\{i:j\}} \cdot P_j^M + (1 - \delta_{\{i:j\}}) \cdot P_j^F \qquad \delta_{\{i:j\}} = \frac{R_{\{i:j\}} - T_j(b)}{R_{\{i:j\}}}$$

$$\begin{aligned} P_{Op}(t) &= P^Y + P^C(t) \\ &= P^Y + P^S + P_{Op}^D(t) \\ &= P^Y + P^S + \sum_{k=1}^{c} \sum_{j=1}^{r} \sum_{i=1}^{n_j} P_{\{i:j\}}^D \cdot M_{k,\{i:j\}}(t) \\ &= P^Y + P^S + \sum_{k=1}^{c} \sum_{j=1}^{r} \sum_{i=1}^{n_j} \left(\delta_{\{i:j\}} \cdot P_j^M + (1 - \delta_{\{i:j\}}) \cdot P_j^F \right) \cdot M_{k,\{i:j\}}(t) \end{aligned}$$

Modeling Power

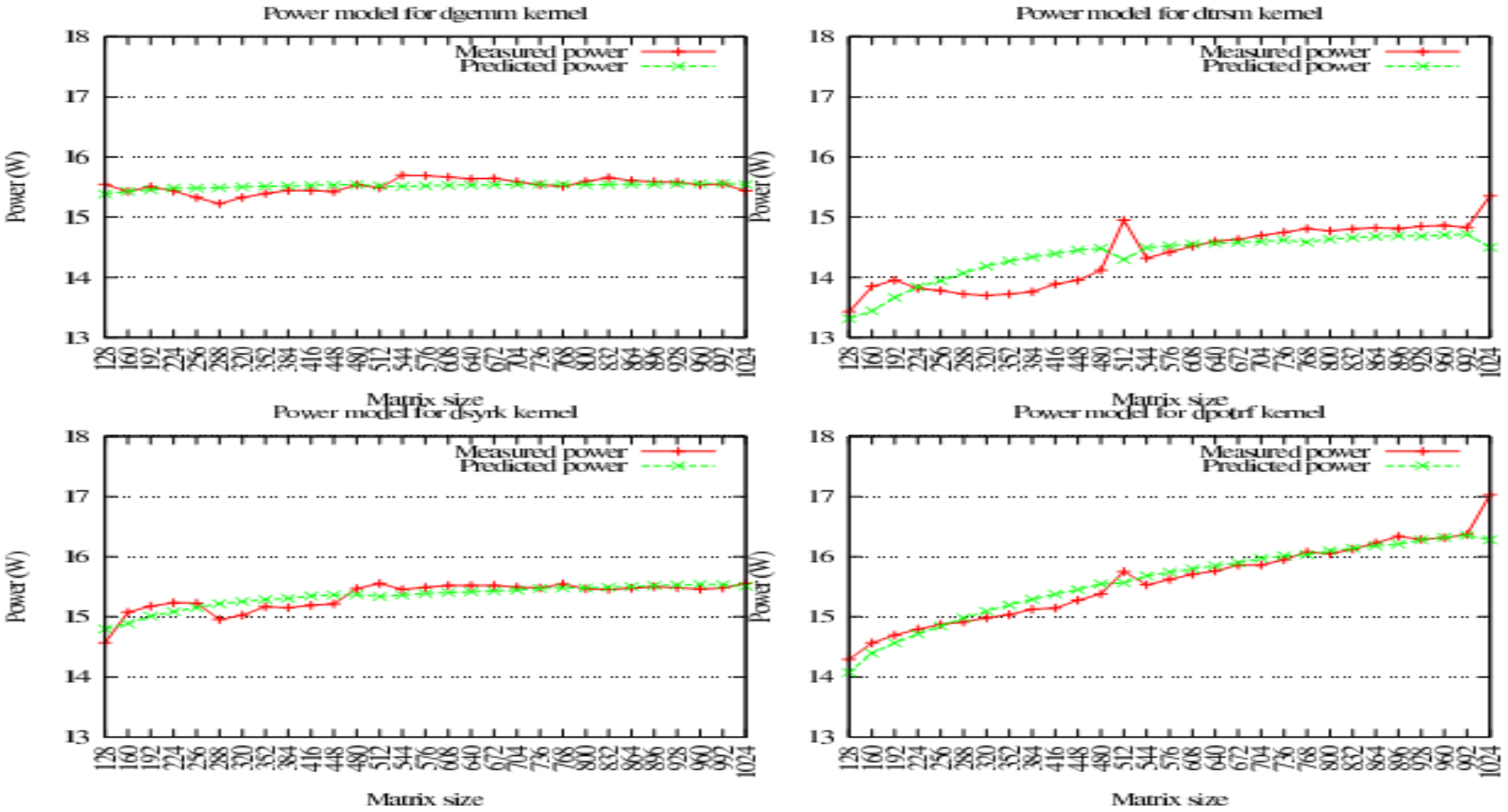
- Task-parallel DLA on multicore and CPU-GPU
  - Accommodate memory contention

$$\begin{pmatrix} \bar{\delta}_{j,128} & (1-\bar{\delta}_{j,128}) \\ \bar{\delta}_{j,160} & (1-\bar{\delta}_{j,160}) \\ \vdots & \vdots \\ \bar{\delta}_{j,1024} & (1-\bar{\delta}_{j,1024}) \end{pmatrix} \begin{pmatrix} P_j^M \\ P_j^F \end{pmatrix} = \begin{pmatrix} \bar{P}^D_{j,128} \\ \bar{P}^D_{j,160} \\ \vdots \\ \bar{P}^D_{j,1024} \end{pmatrix}$$

Task	P_j^M	P_j^F	$\min_b \bar{\delta}_{j,b}$–$\max_b \bar{\delta}_{j,b}$
CHOLESKY FACTORIZATION	13.32	18.72	45–86
TRIANGULAR SOLVE	7.47	15.66	14–28
SYMMETRIC RANK-b UPDATE	12.83	16.00	15–38
MATRIX-MATRIX PRODUCT	14.67	15.70	7–15
LU FACTORIZATION	12.83	17.75	75–95
TRIANGULAR SOLVE	12.12	19.40	55–80
2X1 LU FACTORIZATION	12.54	16.54	33–76
2X1 TRIANGULAR SOLVE	12.53	19.55	81–86
QR FACTORIZATION	15.30	16.88	62–85
APPLY ORTH. TRANSF.	12.10	26.98	76–86
2X1 QR FACTORIZATION	13.91	19.18	65–82
2X1 APPLY ORTH. TRANSF.	6.84	16.72	16–32
BUSY WAIT	0	9.21	–

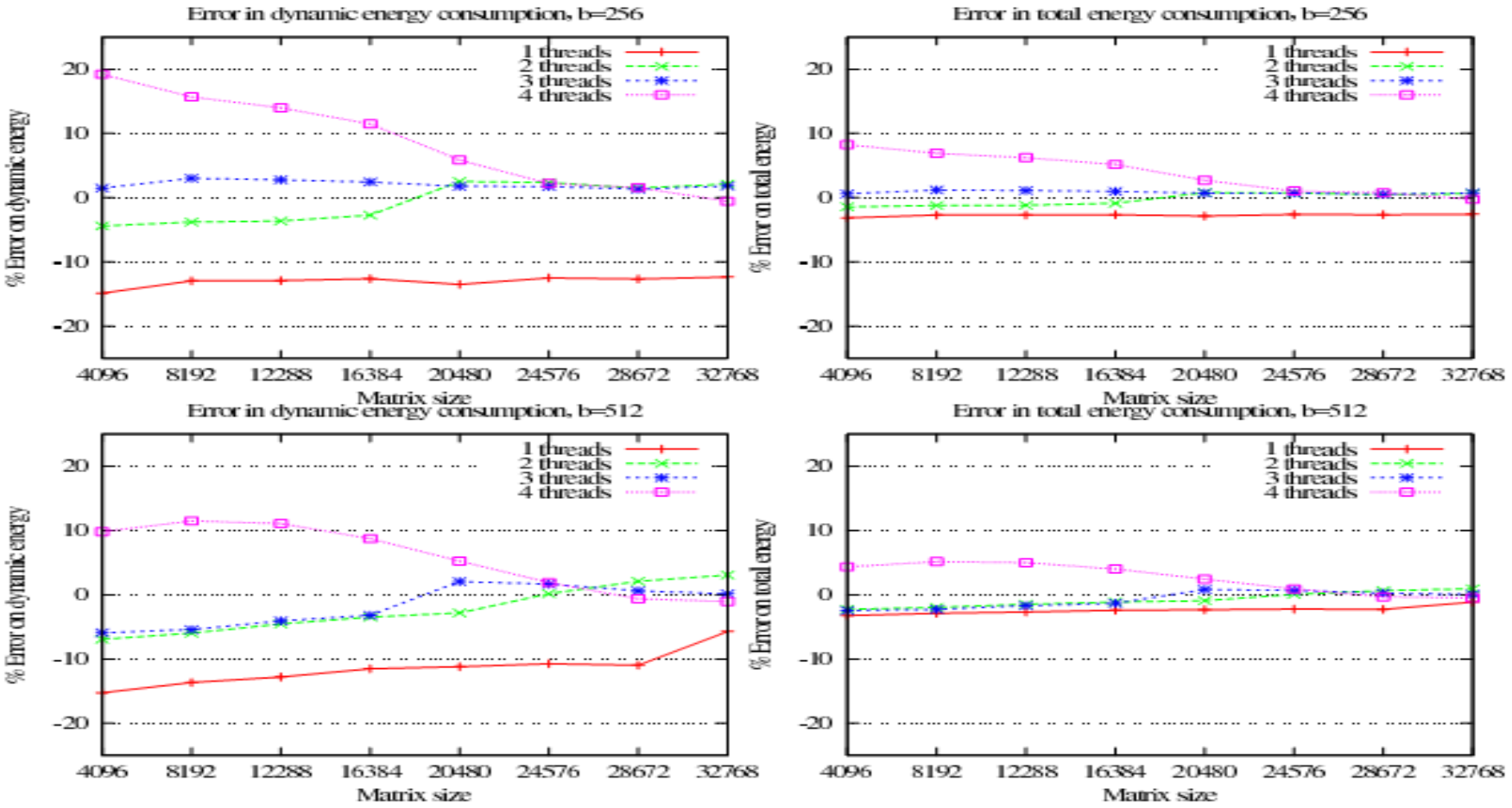
Modeling Power

- Task-parallel DLA on multicore and CPU-GPU



Modeling Power

- Task-parallel DLA on multicore and CPU-GPU



Modeling Power

UNIVERSITAT JAUME·I

- Simple, yet accurate:
  - Dense factorizations (Cholesky, LU, QR)
    - Multicore processors

      "Modeling power and energy consumption of dense matrix factorizations on multicore processors"
      P. Alonso, M. F. Dolz, R. Mayo, E. S. Quintana. CCPE 2013

    - CPU-GPU platforms

      "Enhancing performance and energy consumption of runtime schedulers for dense linear algebra"
      P. Alonso, M. F. Dolz, F. D. Igual, R. Mayo, E. S. Quintana. CCPE 2013 (submitted)

  - ILUPACK on multicore processors

    "Assessing the impact of the CPU power-saving modes on the task-parallel solution of sparse linear systems"
    J. Aliaga, M. Barreda, M. F. Dolz, A. Martín, R. Mayo, E. S. Quintana. Cluster Computing 2013 (submitted)

Outline

- Modeling power
- Saving power in task-parallel appl.
  - ILUPACK for multicore processors
  - CG for hybrid CPU-GPU platforms
- Conclusions

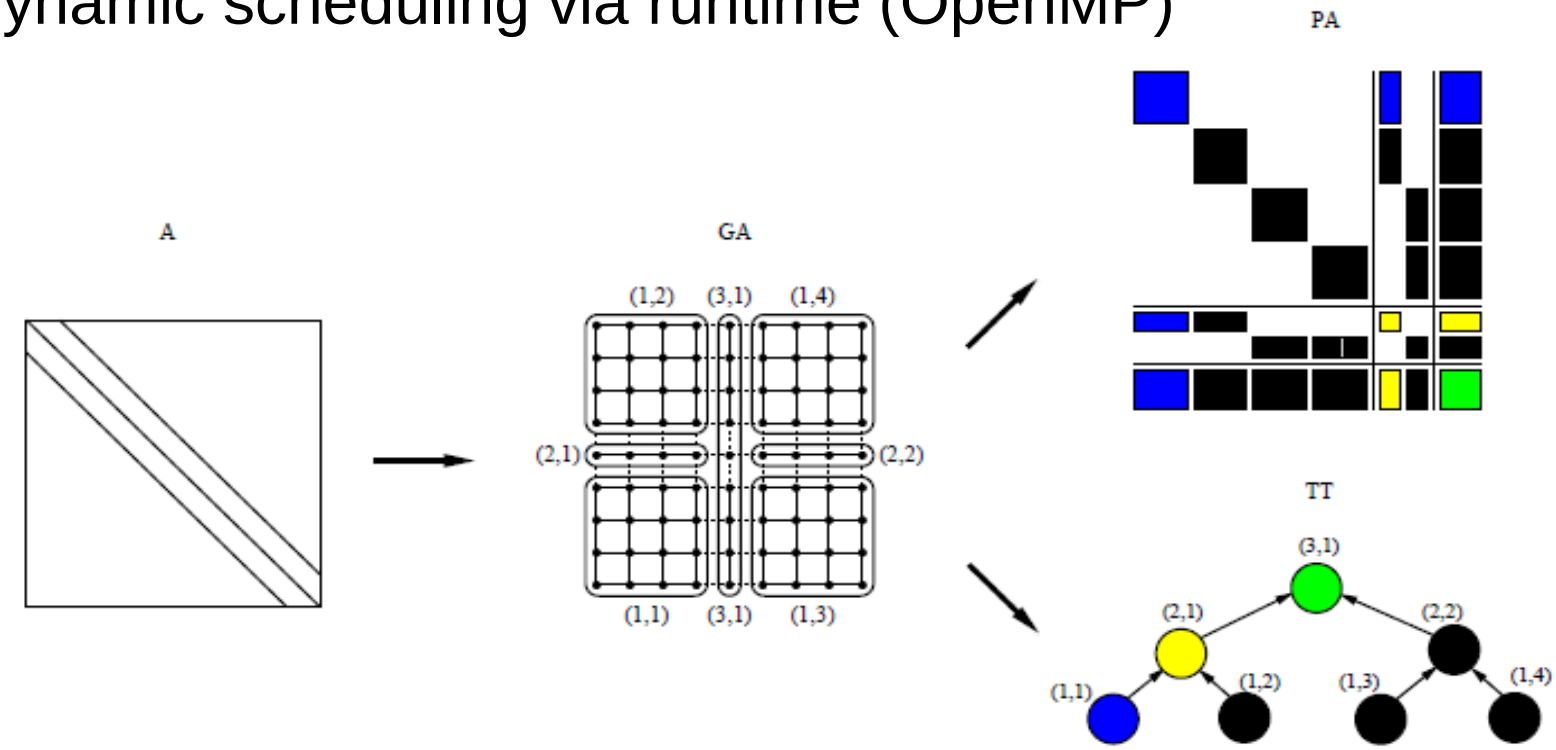
ILUPACK on multicore

UNIVERSITAT JAUME·I

- Incomplete LU Package (http://ilupack.tu-bs.de)
  - Iterative Krylov subspace methods
  - Multilevel ILU preconditioners for general/symmetric/Hermitian positive definite systems
  - Based on inverse ILUs with control over growth of inverse triangular factors
  - Specially competitive for linear systems from 3D PDEs

ILUPACK on multicore
Task parallelism

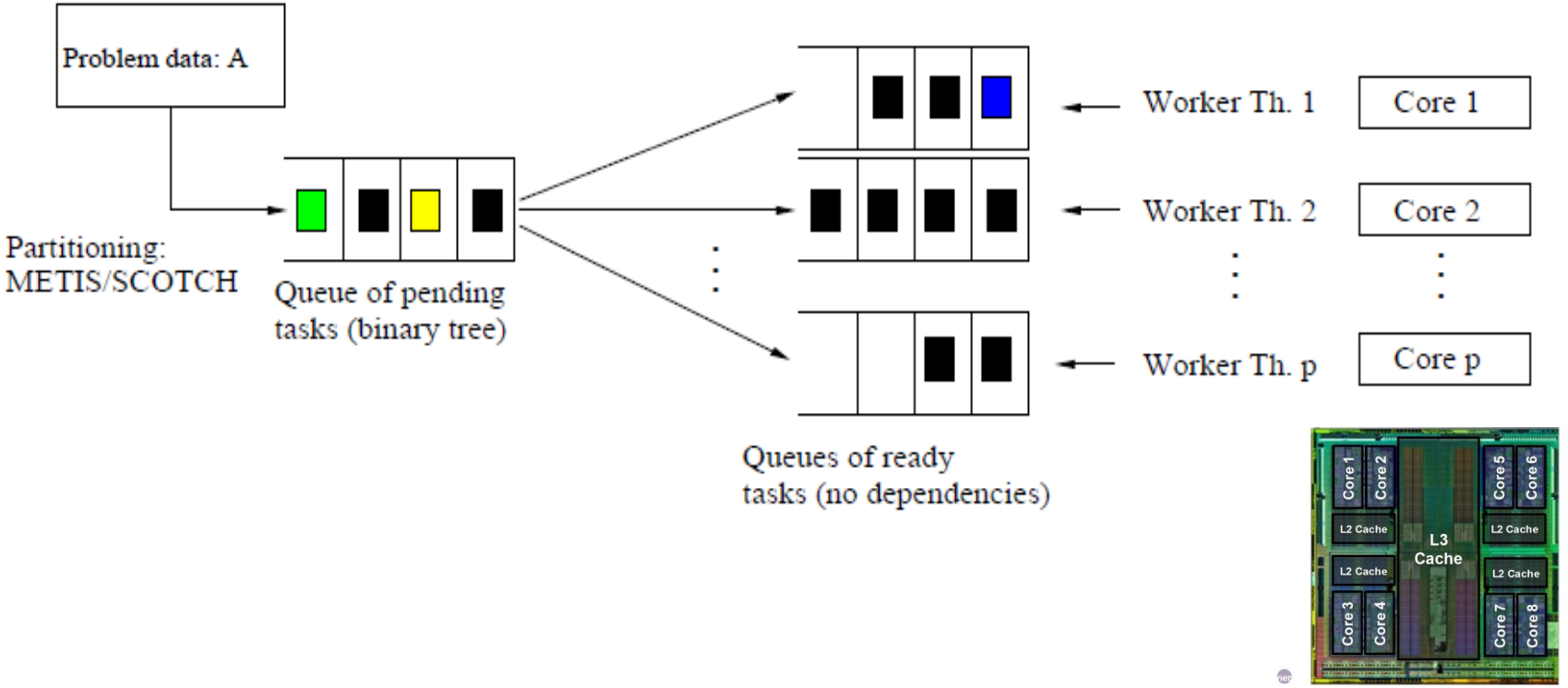
- Multi-threaded parallelism (real s.p.d. systems)
  - Leverage task parallelism
  - Dynamic scheduling via runtime (OpenMP)



ILUPACK on multicore
Task parallelism

UNIVERSITAT JAUME·I

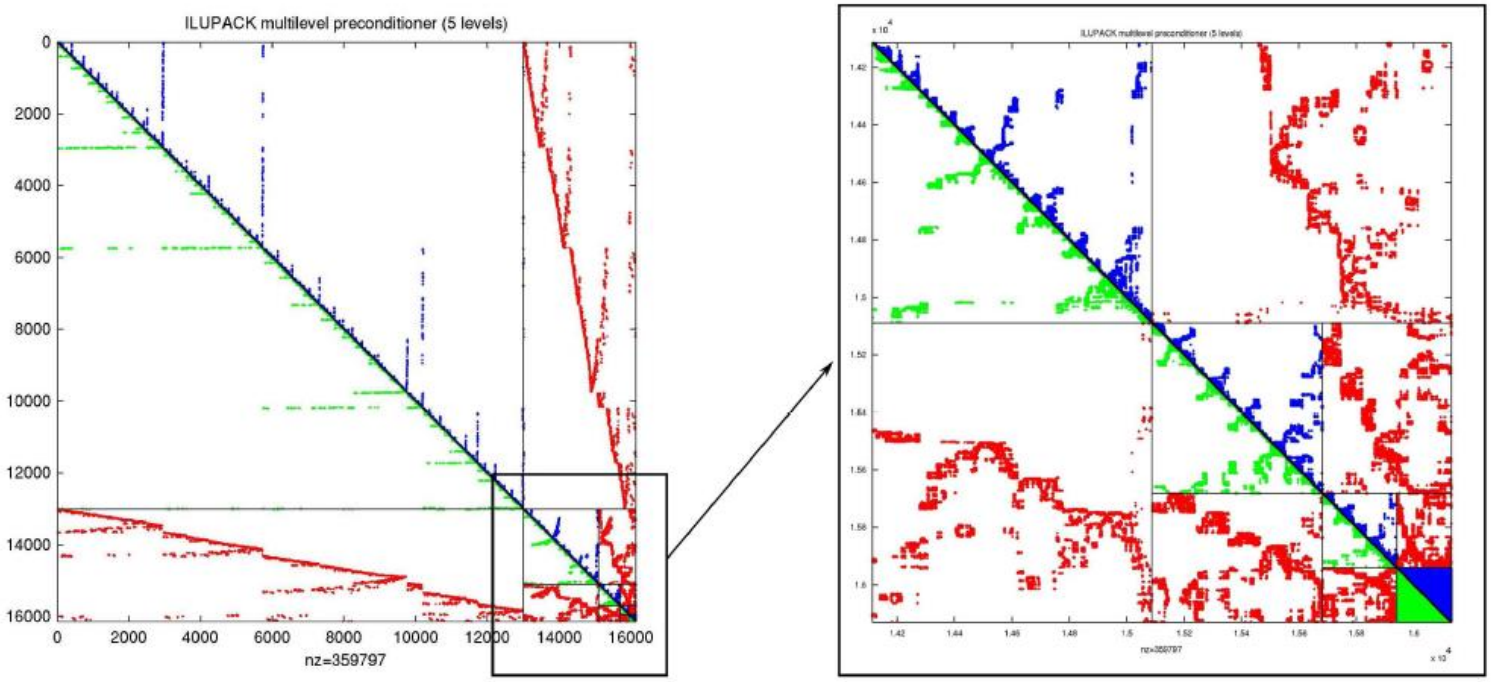
- Run-time in charge of scheduling



"Exploiting thread-level parallelism in the iterative solution of sparse linear systems"
J. I. Aliaga, M. Bollhöfer, A. F. Martín, E. S. Quintana. Parallel Computing, 2011

ILUPACK on multicore
Experimental setup

- Sparse linear system benchmark
  - Laplacian equation $-\Delta u = f$ in a 3D unit cube $\Omega = [0,1]^3$
  - Linear system $Au = b$ with $A \rightarrow n \times n$, $n = 252^3 \approx$ 16 million unknowns and 111 millions of nonzero entries



ILUPACK on multicore
Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

Platform	P-state, P_i	V_i	f_i
WT_AMD	P0	1.23	2.00
	P1	1.17	1.50
	P2	1.12	1.20
	P3	1.09	1.00
	P4	1.06	0.80

- DVFS = P-states (see ACPI standard)
- Moving to a higher P-state results in ↓power
- ↓Power = ↓Energy?
- For a compute-bounded operation, f_i is linear to time^{-1}
- In principle, for a memory-bounded operation (ILUPACK), reducing f_i should have a minor impact on performance

ILUPACK on multicore
Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

- 1st attempt: ~~Dynamic~~ Static voltage-frequency scaling

P-state P_i	T_i	$\bar{P}_i^T$	E_i	ΔT_i	$\Delta \bar{P}_i^T$	ΔE_i
P_0	34.06	282.87	9,634.78	–	–	–
P_1	43.57	235.64	10,267.72	21.88	-16.69	6.53
P_2	54.48	210.86	11.478.79	59.91	-25.45	19.20
P_3	61.58	197.01	12.132.79	80.73	-30.35	25.87
P_4	76.50	186.86	14,295.18	124.47	-33.94	48.28

Why?

ILUPACK on multicore
Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

- 1st attempt: ~~Dynamic~~ Static voltage-frequency scaling

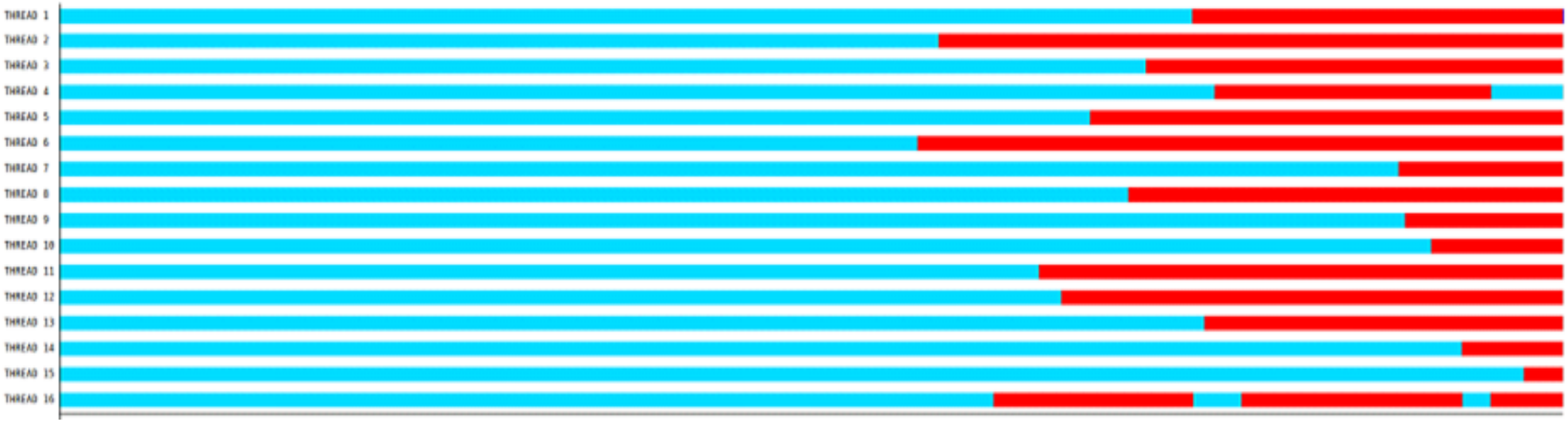
P-state P_i	Vcc_i	f_i	T_i	ΔT_i	BW_i	ΔBW_i
P_0	1.23	2.00	34.06	–	30.29	–
P_1	1.17	1.50	43.57	21.88	24.63	-18.67
P_2	1.12	1.20	54.48	59.91	20.46	-32.44
P_3	1.09	1.00	61.58	80.73	17.48	-42.30
P_4	1.06	0.80	76.50	124.47	14.00	-53.77

  - Combined effect of linear decrease of CPU performance and memory bandwidth!
  - Decrease of P^s_i ($P_0 \rightarrow P_2$: -21.47%), decrease of P^D_i ($P_0 \rightarrow P_3$: -60.73%) but P^y_i does not change!

ILUPACK on multicore
Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

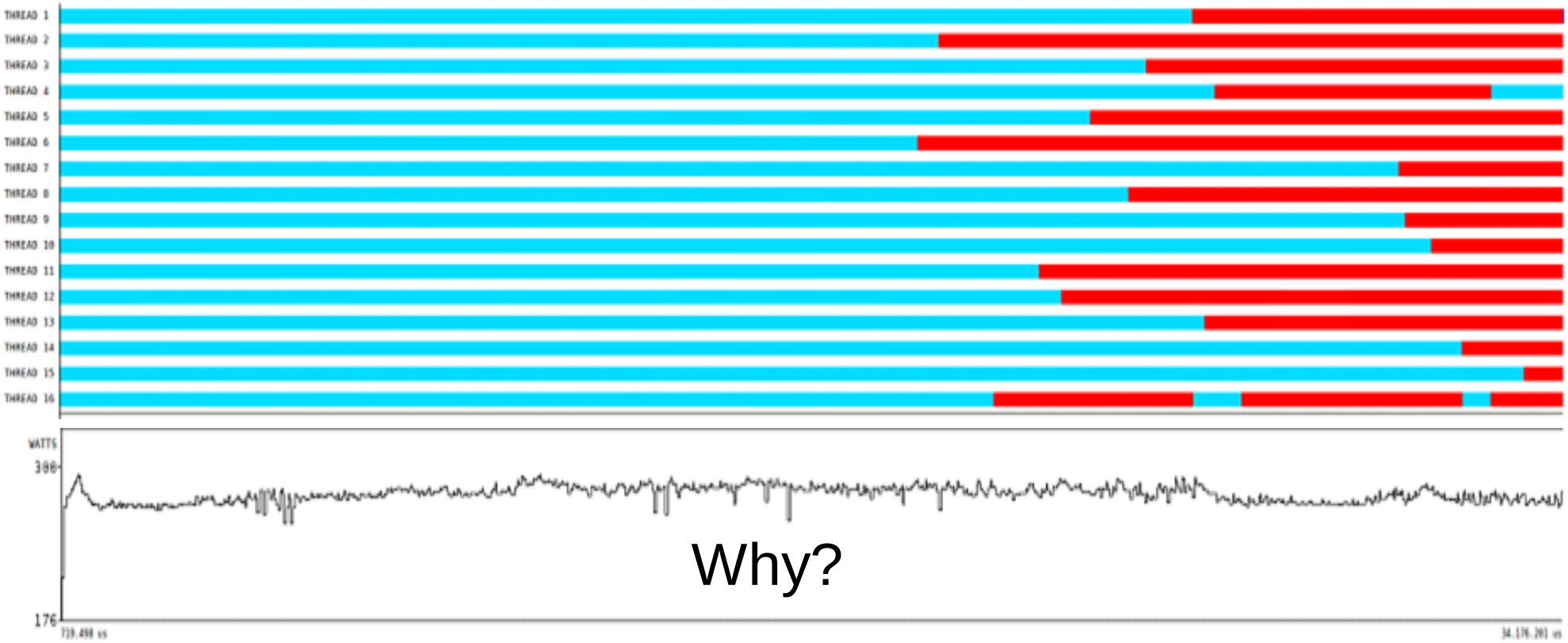
- 2nd attempt: DVFS during idle periods



ILUPACK on multicore Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

- 2nd attempt: DVFS during idle periods

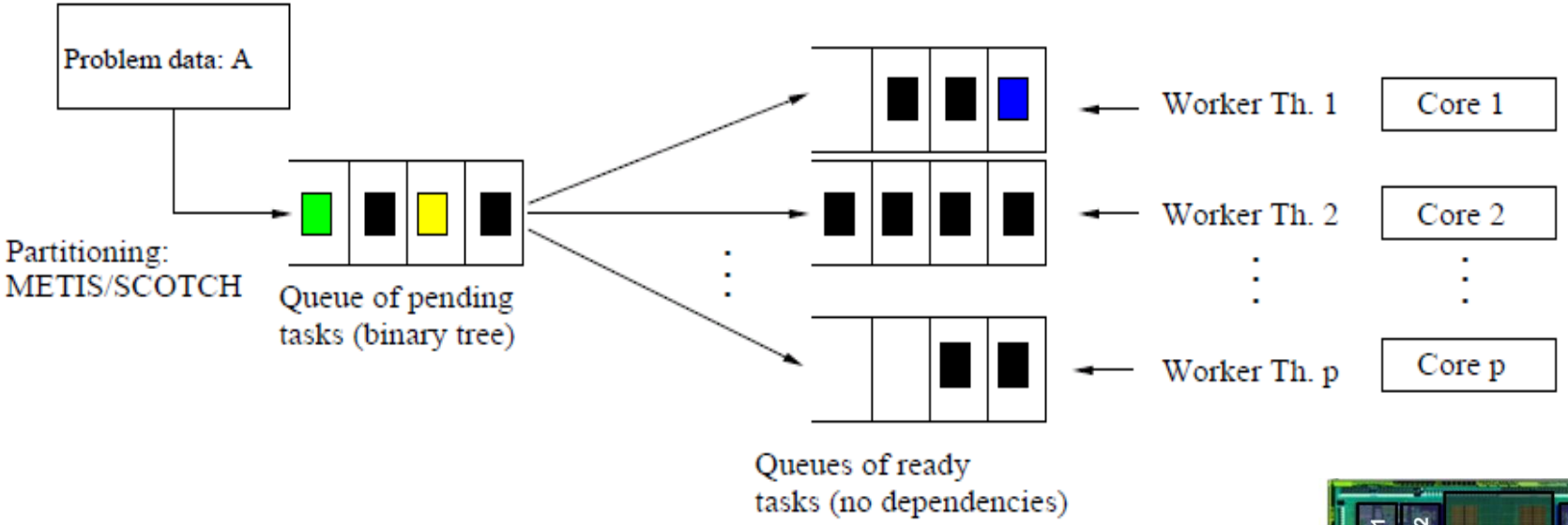


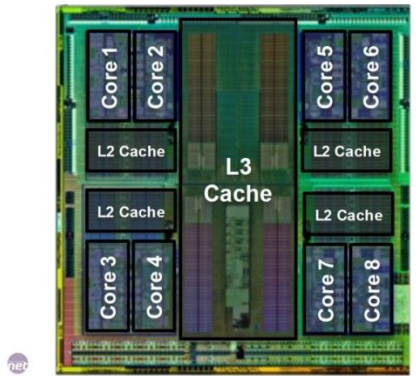
Why?

ILUPACK on multicore Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

- 2nd attempt: DVFS during idle periods

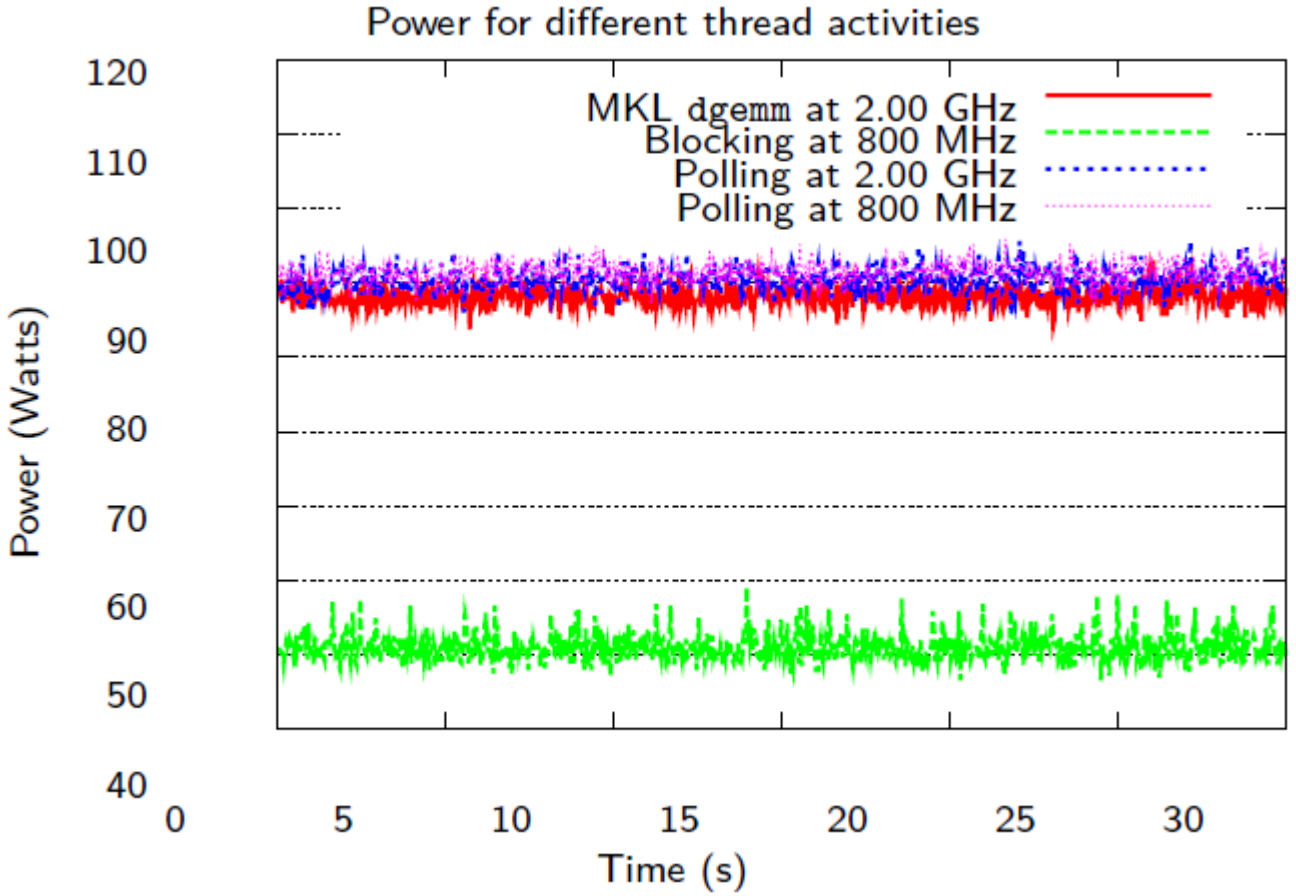




ILUPACK on multicore
Leveraging P-states (AMD)

UNIVERSITAT JAUME·I

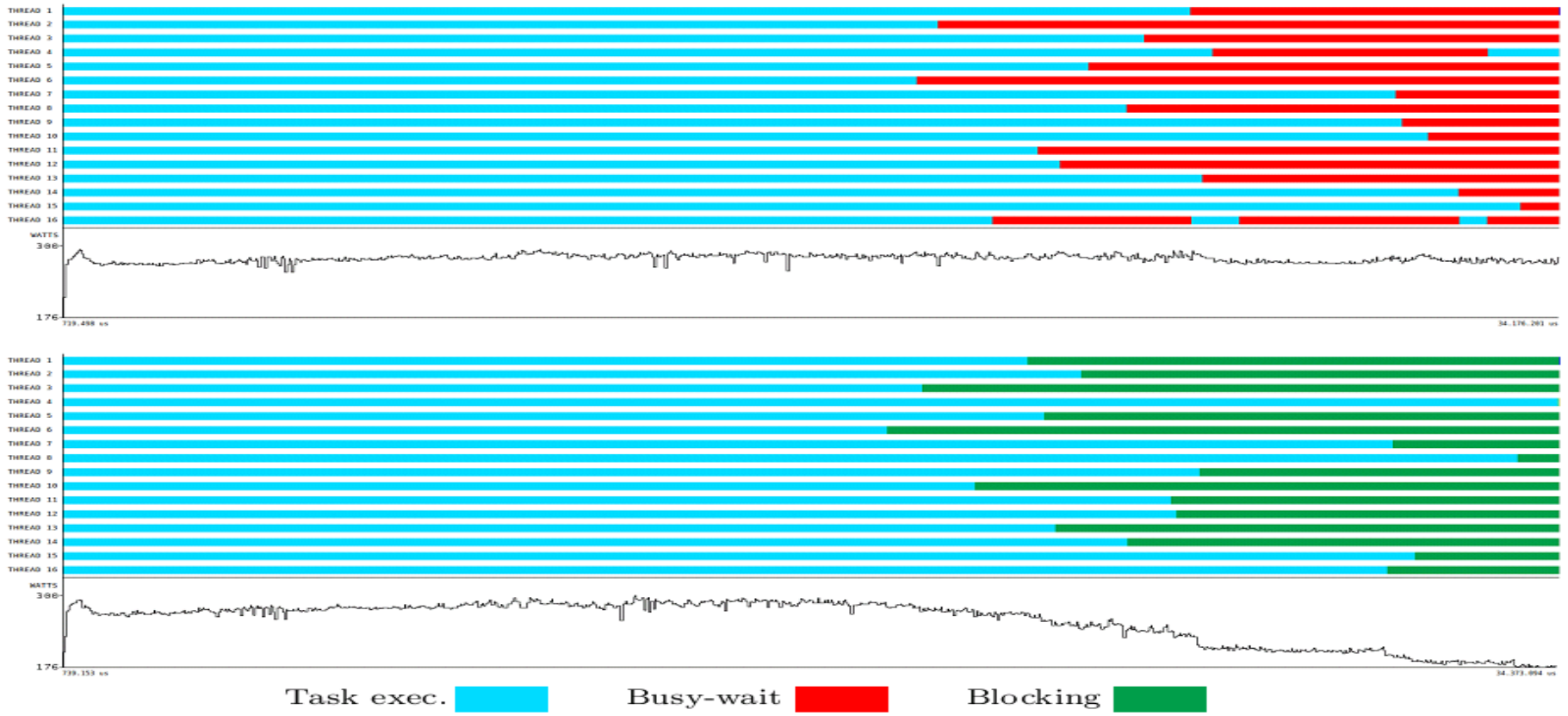
- Active polling for work…



ILUPACK on multicore
Leveraging P- and C-states (AMD)

UNIVERSITAT JAUME·I

- 3rd attempt: DVFS and idle-wait



ILUPACK on multicore
Leveraging P- and C-states (AMD)

UNIVERSITAT JAUME·I

- 3rd attempt: DVFS and idle-wait:
  - Savings of 6.92% of total energy
  - Negligible impact on execution time

- …but take into account that
  - Idle time: 23.70%
  - Dynamic power: 32.32%
  - Upper bound of savings: $39.32 \cdot 0.2370 = 9.32\%$

ILUPACK on multicore
Leveraging P-states (Intel)

P-state	VCC_i	f_i	BW_i	ΔBW_i
P_0	1.04	2.00	12.72	–
P_1	0.98	1.73	12.61	-0.87
P_2	0.95	1.60	12.55	-1.34
P_3	1.01	1.87	12.58	-1.10

DVFS

	P_i	T_i	$\bar{P}_i^T$	E_i	ΔT_i	$\Delta \bar{P}_i^T$	ΔE_i
ILU	P_0	56.43	135.17	7,627.97	–	–	–
	P_1	59.06	127.96	7,557.87	4.67	-5.33	-0.92
	P_2	62.93	121.99	7,676.98	11.52	-9.75	0.64
	P_3	67.05	116.22	7,792.77	18.82	-18.82	2.16
Solve	P_0	148.94	155.27	23,123.99	–	–	–
	P_1	148.52	151.07	22,434.73	-0.28	-2.70	-2.98
	P_2	154.86	145.11	22,469.38	3.97	-6.55	-2.83
	P_3	159.08	138.50	22,033.14	6.81	-10.80	-4.72

ILUPACK on multicore
Leveraging P- and C-states (Intel)

UNIVERSITAT JAUME·I

Average reduction: 9.5% for LU and 6.5% for Solve

DVFS

	P_i	T_i	$\bar{P}_i^T$	E_i	ΔT_i	$\Delta \bar{P}_i^T$	ΔE_i
ILU	P_0	56.43	135.17	7,627.97	–	–	–
	P_1	59.06	127.96	7,557.87	4.67	-5.33	-0.92
	P_2	62.93	121.99	7,676.98	11.52	-9.75	0.64
	P_3	67.05	116.22	7,792.77	18.82	-18.82	2.16
Solve	P_0	148.94	155.27	23,123.99	–	–	–
	P_1	148.52	151.07	22,434.73	-0.28	-2.70	-2.98
	P_2	154.86	145.11	22,469.38	3.97	-6.55	-2.83
	P_3	159.08	138.50	22,033.14	6.81	-10.80	-4.72

DVFS+idle-wait

	P_i	T_i	$\bar{P}_i^T$	E_i	ΔT_i	$\Delta \bar{P}_i^T$	ΔE_i
ILU	P_0	55.96	122.83	6,872.82	–	–	–
	P_1	59.50	116.52	6,932.86	6.34	-5.14	0.87
	P_2	62.37	112.42	7,012.13	11.46	-8.47	2.03
	P_3	66.84	107.34	7,174.81	19.46	-12.61	4.39
Solve	P_0	147.40	146.63	21,594.05	–	–	–
	P_1	148.41	143.18	21,241.34	0.68	-2.35	-1.63
	P_2	151.39	138.96	21,037.62	2.71	-5.23	-2.58
	P_3	158.28	132.65	20,992.39	7.38	-9.53	-2.79

Outline

- Modeling power
  - ILUPACK for multicore processors
- Saving power in task-parallel appl.
  - ILUPACK for multicore processors
  - CG for hybrid CPU-GPU platforms
- Conclusions

The CG method on CPU-GPU

UNIVERSITAT JAUME·I

- Leveraging P-states on CPU-GPU platforms?
  - Apply DVFS to the CPU while computation proceeds on the GPU?

- Leveraging C-states on CPU-GPU platforms?
  - What is the CPU doing while computation proceeds on the GPU?

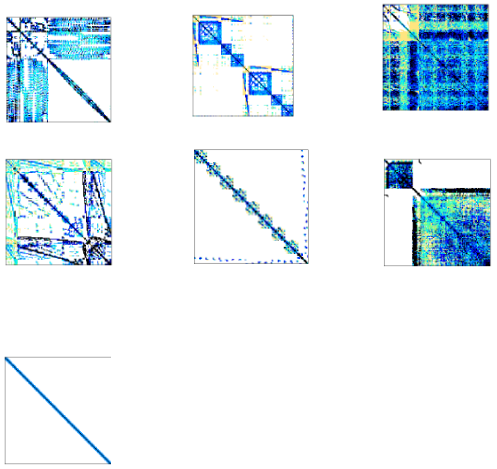
The CG method on CPU-GPU
Experimental setup

UNIVERSITAT JAUME·I

- Sandy:
  - Intel i7-3770K, 16GB
  - NVIDIA GeForce GTX480

- Cases from two matrix collections

Source	Matrix	#nonzeros (n_z)	Size (n)	n_z/n
UFMC	AUDIKW_1	77,651,847	943,645	82.28
	BMWCRA1	10,641,602	148,770	71.53
	CRANKSEG_2	14,148,858	63,838	221.63
	F1	26,837,113	343,791	78.06
	INLINE_1	38,816,170	503,712	77.06
	LDOOR	42,493,817	952,203	44.62
Laplace	A100	6,940,000	1,000,000	6.94
	A126	13,907,370	2,000,376	6.94
	A159	27,986,067	4,019,679	6.94
	A200	55,760,000	8,000,000	6.94
	A252	111,640,032	16,003,001	6.94



The CG method on CPU-GPU
Basic implementation

UNIVERSITAT JAUME·I

- CG: Sparse matrix-vector (SpMV) + CUBLAS

```
while( ( k < maxiter ) && ( res > epsilon ) ){
  SSpMV <<<Gs,Bs>>> ( n, rowA, colA, valA, d, z );
  tmp = cublasSdot ( n, d, 1, z, 1 );
  rho = beta / tmp;
  gamma = beta;
  cublasSaxpy (n,  rho, d, 1, x, 1 );
  cublasSaxpy (n, -rho, z, 1, r, 1 );
  beta = cublasSdot( n, r, 1, r, 1 );
  alpha = beta / gamma;
  cublasSscal (n, alpha, d, 1 );
  cublasSaxpy (n, one, r, 1, d, 1 );
  res = sqrt( beta );
  k++;
} // end-while
```

The CG method on CPU-GPU
Basic implementation

- CG: Sparse matrix-vector (SpMV) + CUBLAS

```
while( ( k < maxiter ) && ( res > epsilon ) ){
  SSpMV <<<Gs,Bs>>> ( n, rowA, colA, valA, d, z );
  tmp = cublasSdot ( n, d, 1, z, 1 );
  rho = beta / tmp;
  alpha = beta / gamma;
  cublasSscal (n, alpha, d, 1 );
  cublasSaxpy (n, one, r, 1, d, 1 );
  res = sqrt( beta );
  k++;
} // end-while
```

Leveraging P-states:

- Basically all computation performed on the GPU
- Apply static VFS to reduced power in CPU!

The CG method on CPU-GPU
Basic implementation

- CG: Sparse matrix-vector (SpMV) + CUBLAS

```
while( ( k < maxiter ) && ( res > epsilon ) ){
  SSpMV <<<Gs,Bs>>> ( n, rowA, colA, valA, d, z );
  tmp = cublasSdot ( n, d, 1, z, 1 );
  rho = beta / tmp;
  res = sqrt( beta );
  k++;
} // end-while
```

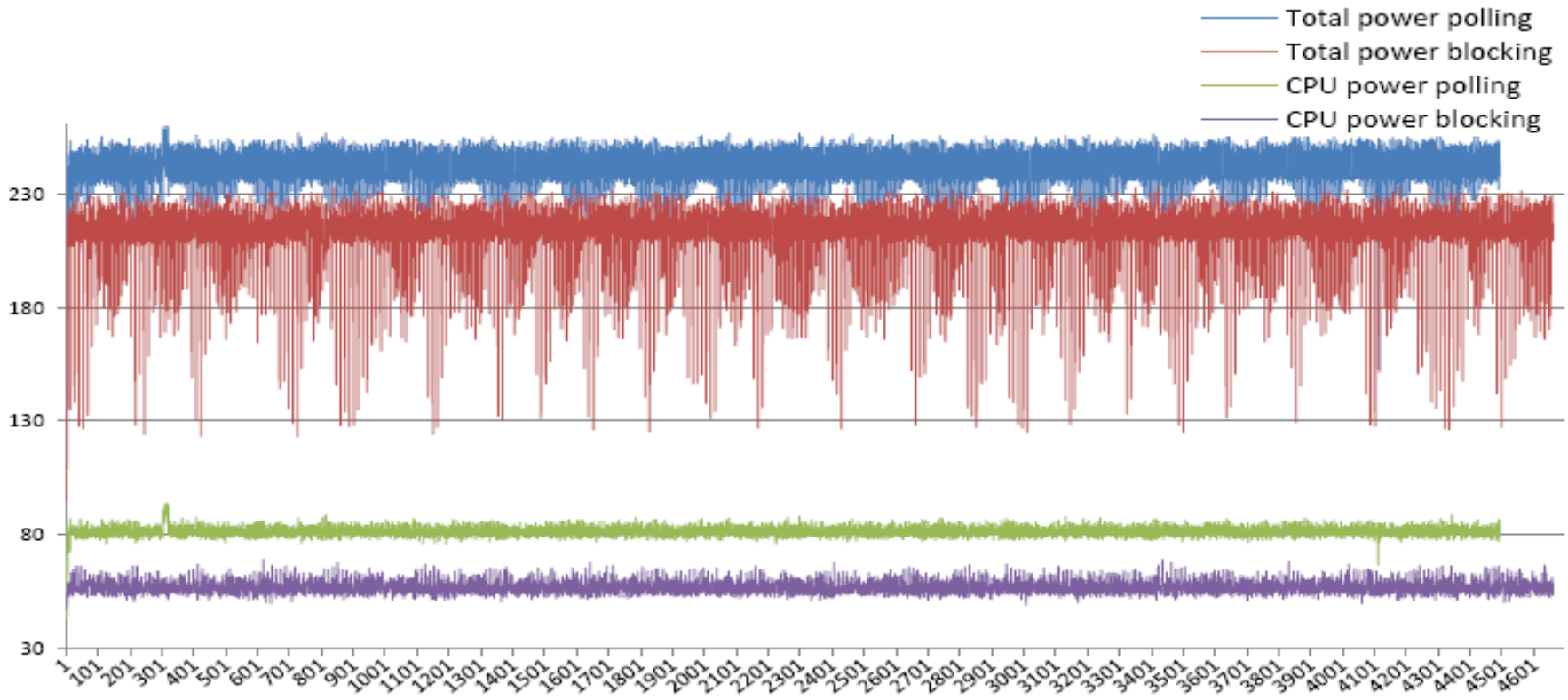
Leveraging C-states:

- What is the CPU doing while computation proceeds on the GPU?
- CUDA offers polling (active-wait) vs blocking (idle-wait) operation modes

The CG method on CPU-GPU
Basic implementation

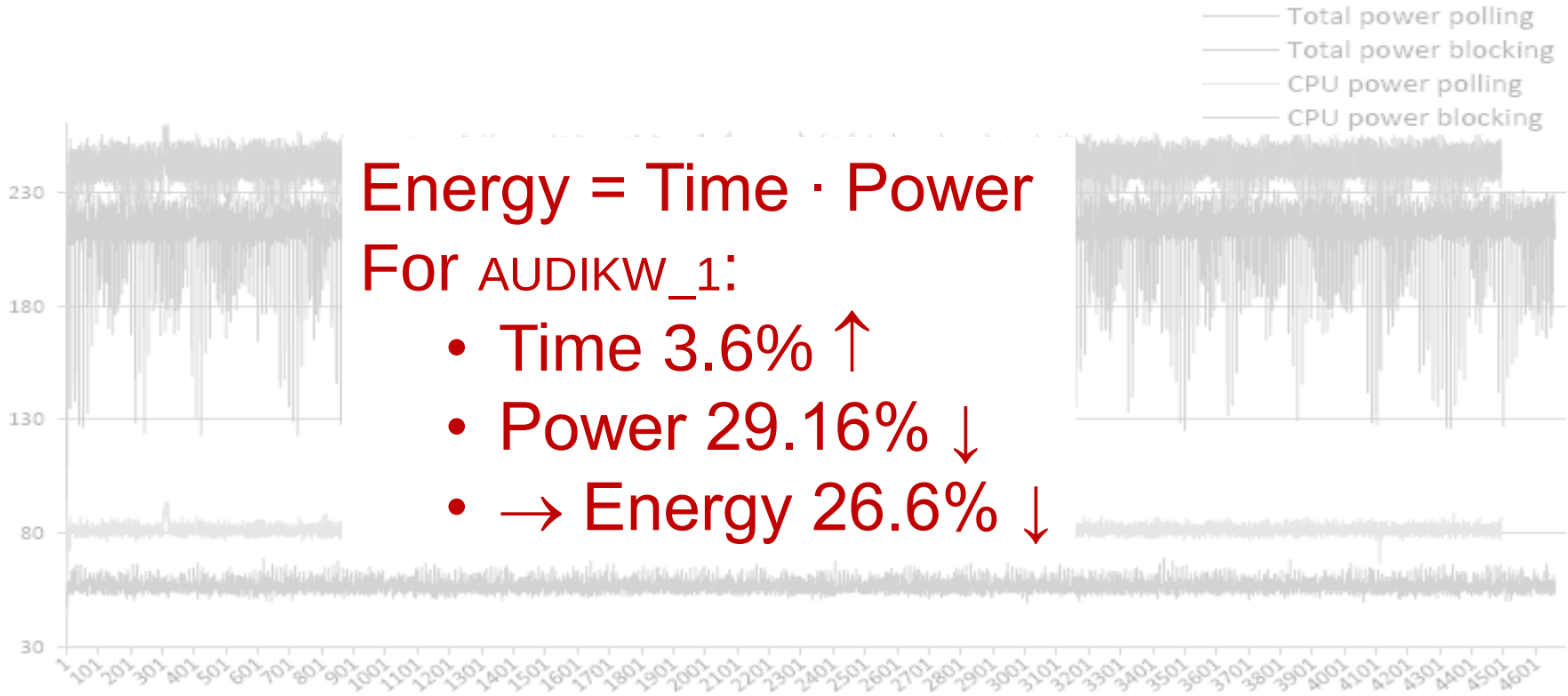
UNIVERSITAT JAUME·I

- Trading off energy for time: variations of CUDA blocking mode w.r.t. CUDA polling mode



The CG method on CPU-GPU
Basic implementation

- Trading off energy for time: variations of CUDA blocking mode w.r.t. CUDA polling mode



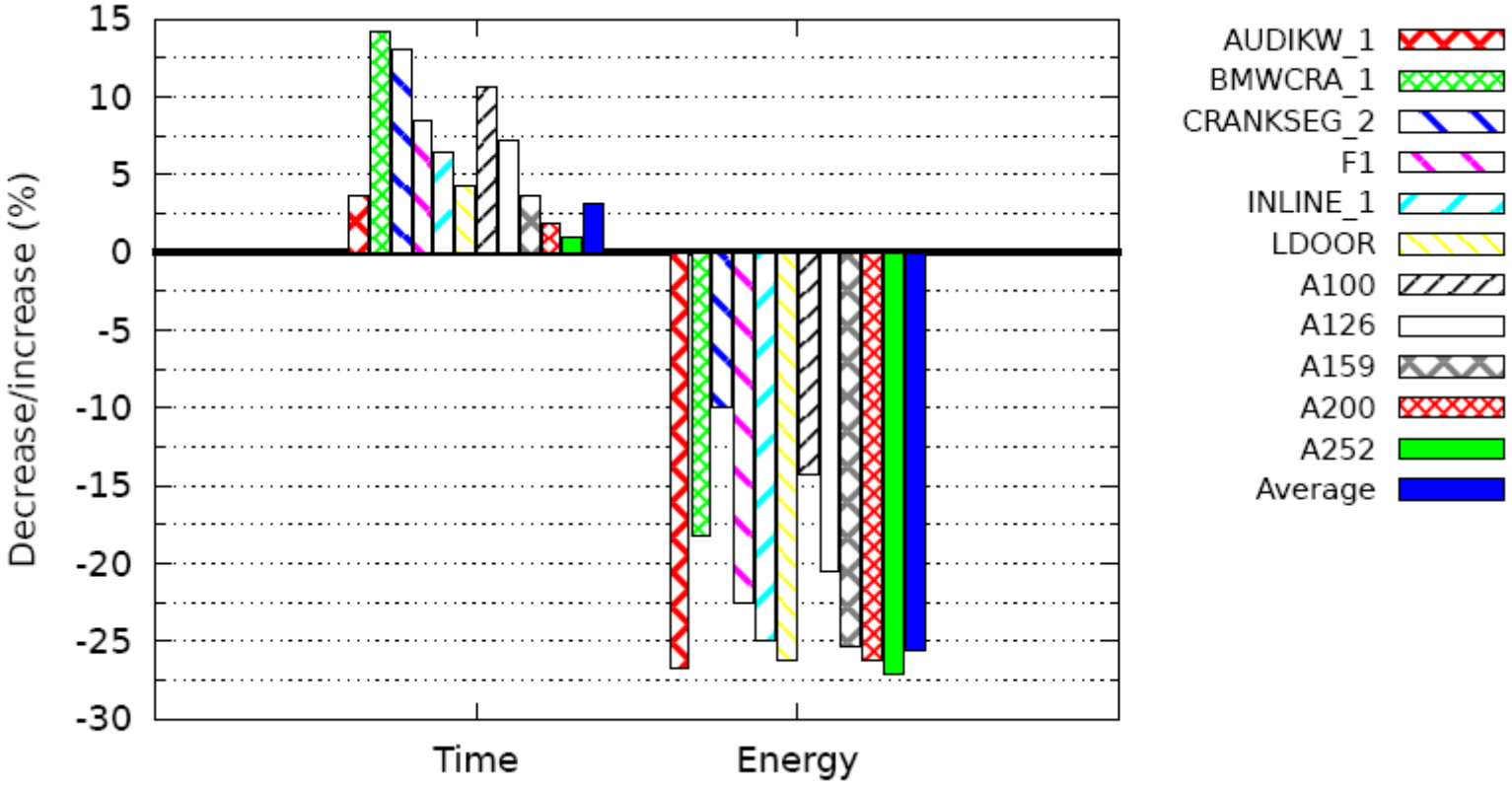
Energy = Time · Power

For AUDIKW_1:

- Time 3.6% ↑
- Power 29.16% ↓
- → Energy 26.6% ↓

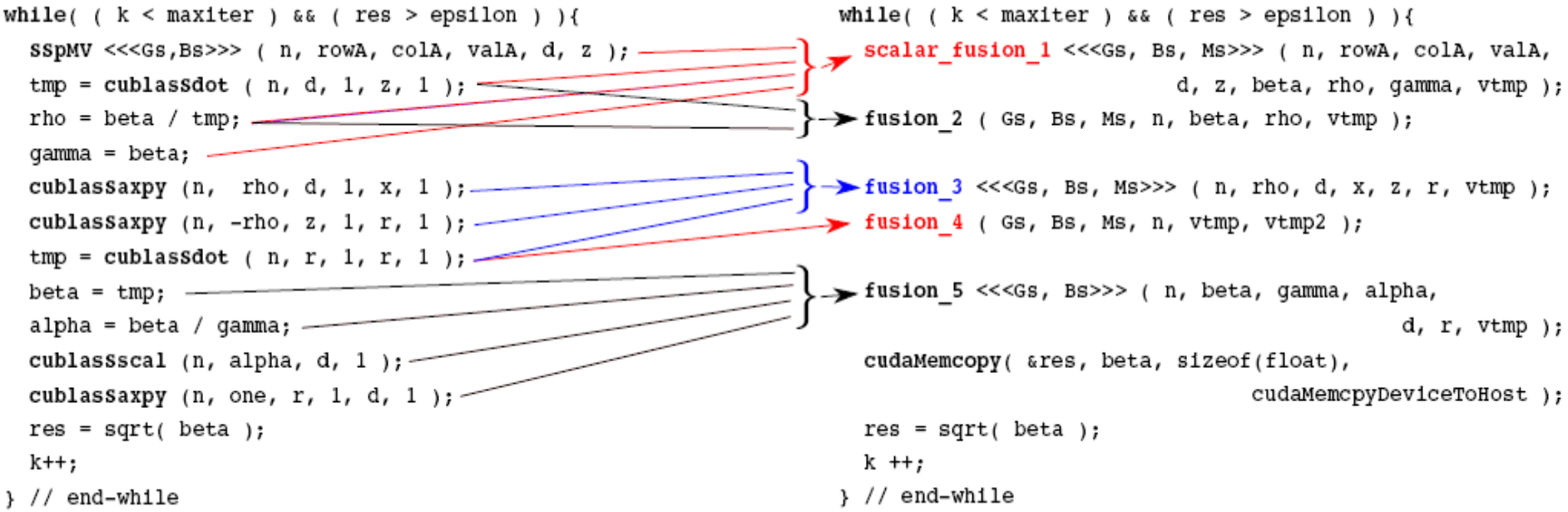
The CG method on CPU-GPU
Basic implementation

- Trading off energy for time: variations of CUDA blocking mode w.r.t. CUDA polling mode



The CG method on CPU-GPU
Merged implementation

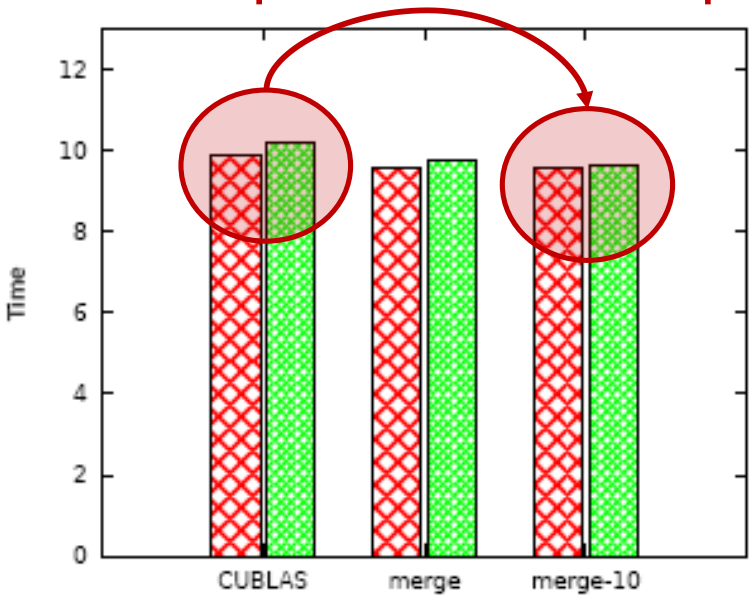
- Can we attain polling performance and blocking energy advantage?
- Requires a reformulation of CG (merge kernels)

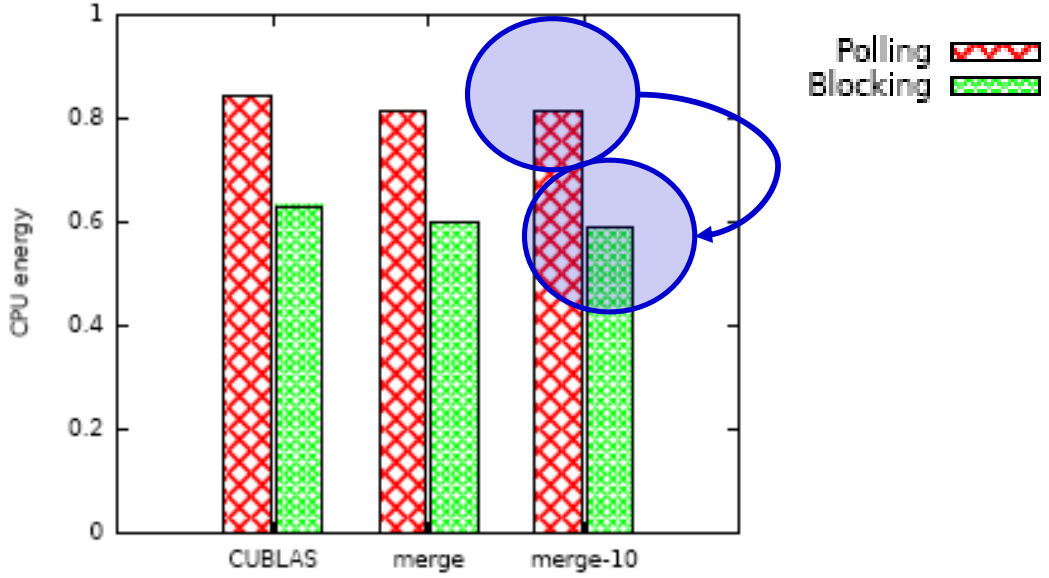


The CG method on CPU-GPU Merged implementation

- Time vs. CPU energy

Maintain performance of polling…





…while leveraging energy-efficiency of C-states+idle-wait

Performance and energy consumption Summary

"*Do nothing, efficiently…*" (V. Pallipadi, A. Belay)

or

"*Doing nothing well*" (D. E. Culler)

Acknowledgments

A. F. Martín

H. Anzt

J. I. Aliaga, M. Barreda, M. F. Dolz, R. Mayo, J. Pérez, E. S. Quintana-Ortí

P. Alonso

Acknowledgments

- EU FP7 318793 Project
  "EXA2GREEN. Energy-Aware Sustainable Computing on Future Technology - Paving the Road to Exascale Computing"

Exa2Green
energy-aware numerics

- Project CICYT TIN2011-23283
  "PA-HPC. Power-Aware High Performance Computing"

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD