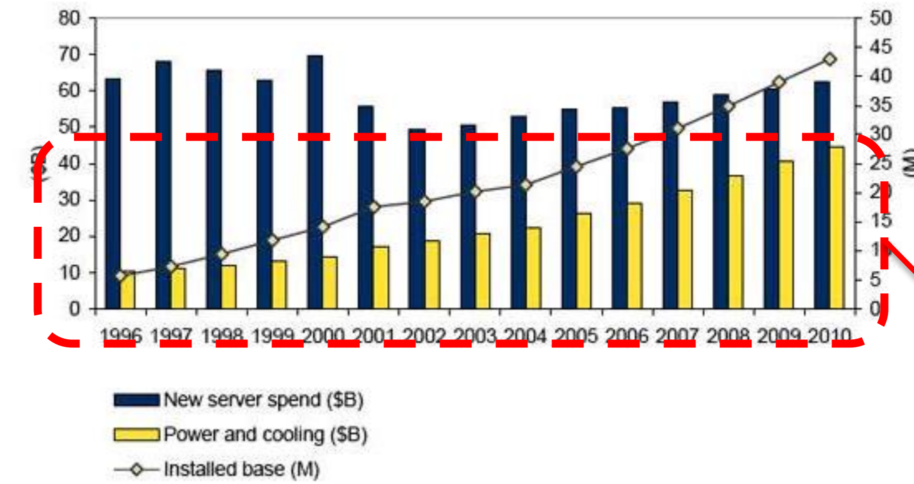
Model predictive control and self-learning of thermal models for multi-core platforms*

Luca Benini
Luca.benini@unibo.it

*Work supported by Intel Labs Braunschweig

The Power Crisis

Worldwide Cost to Power and Cool Server Installed Base, 1996–2010



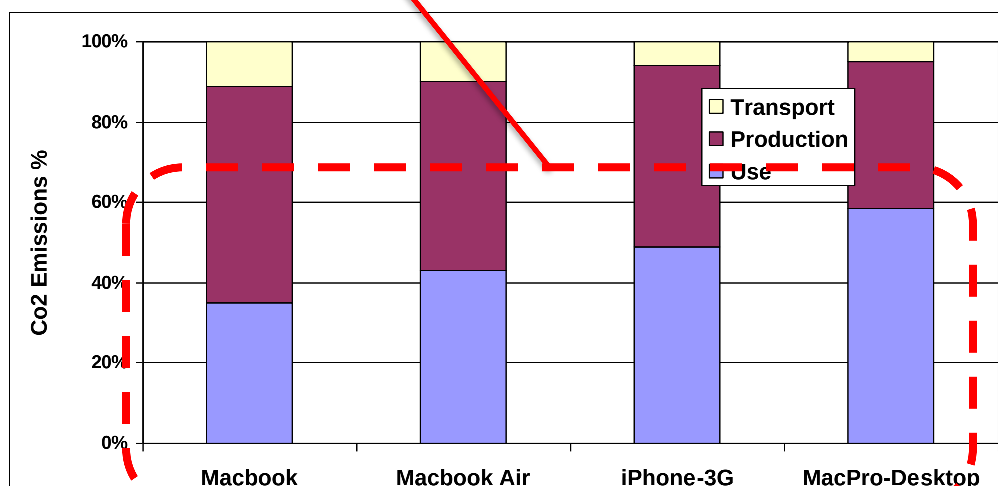
Source: IDC, 2007

Data center, HPC,server

mobile clients

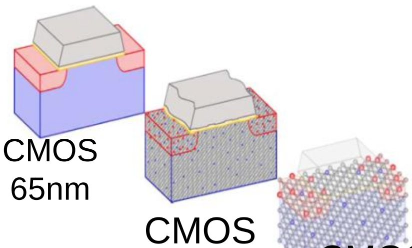
Run-time power
~50% of ICT environmental impact and cost!

M.Pedram [CADS10]



The Thermal Crisis

- Never-ending shrinking: smaller, faster…



Free lunch?!

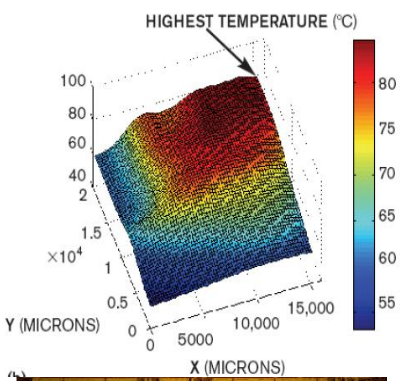
[Cell Multi-Processor]

Not really…

“Cool” chips, “hot” applications

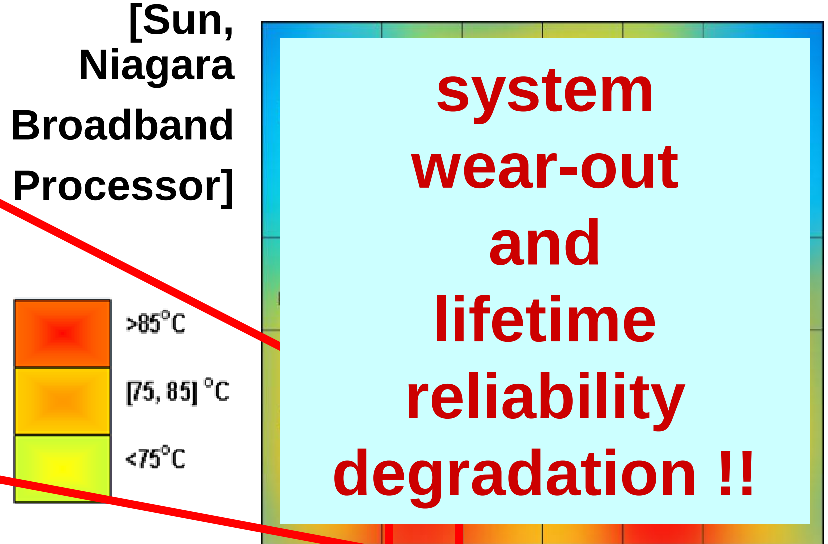
- Multi-Processor SoC possible
- Thermal issues: hot-spots, thermal gradients…

[Sun, 1.8 GHz Sparc v9 Microproc]



[Coskun et al ‘07, UCSD]

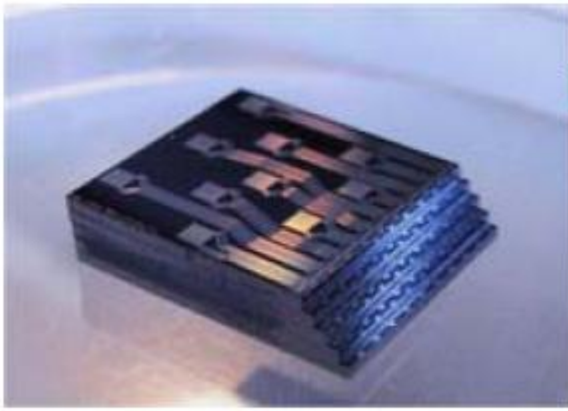
[Sun, Niagara Broadband Processor]



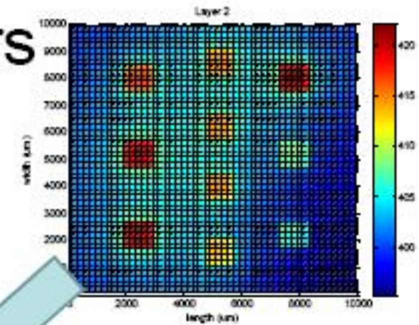
system wear-out and lifetime reliability degradation !!

3D-SoCs are even worse

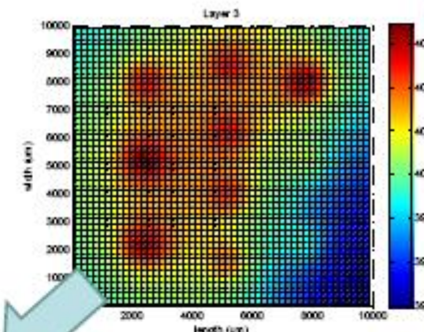
5-tier 3D stack: 10 heat sources and sensors



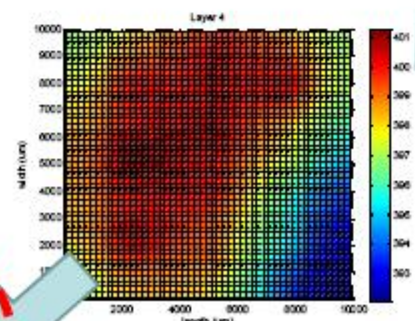
Inject between 4W – 1.5W



2nd Tier

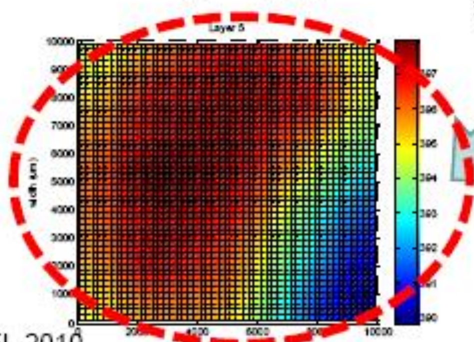


3rd Tier



4th Tier

5th Tier

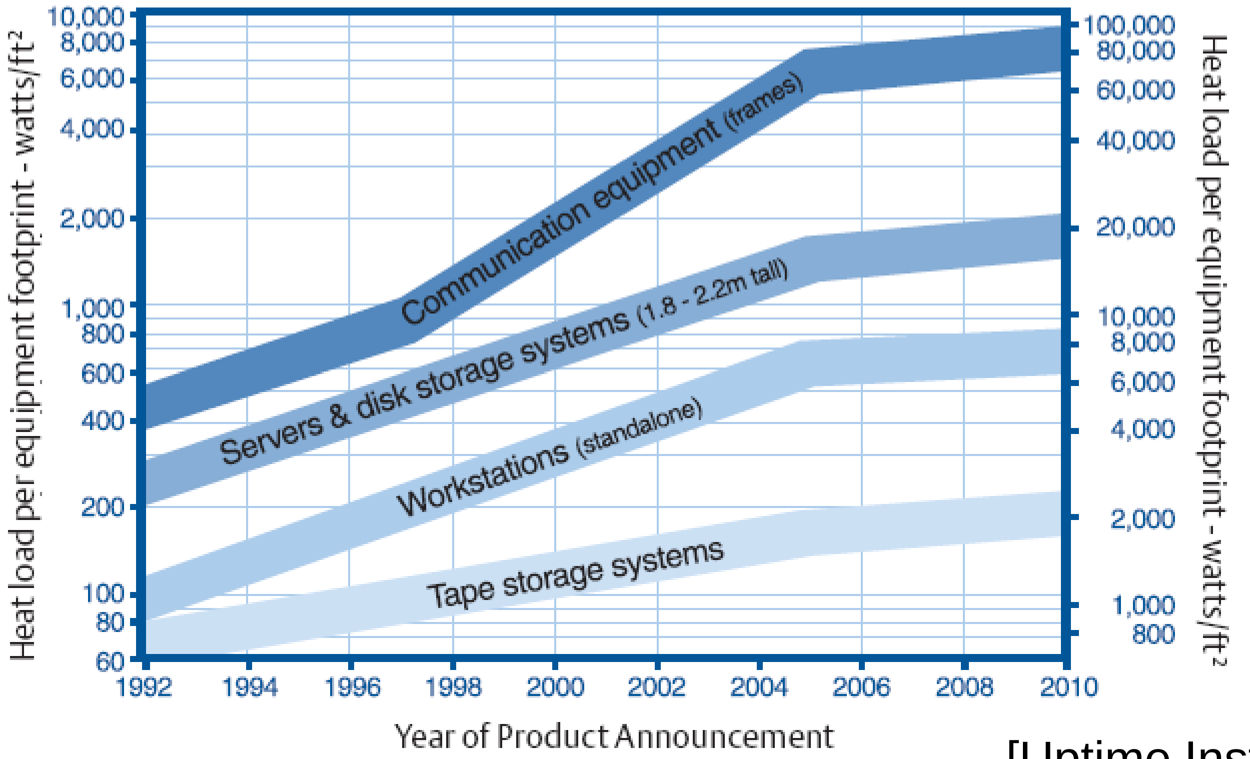


Large and non-uniform heat propagation!
(up to 130º C on top tier)

ESL/EPFL 2010

A System-level View

- Heat density trend 2005-2010 (systems)

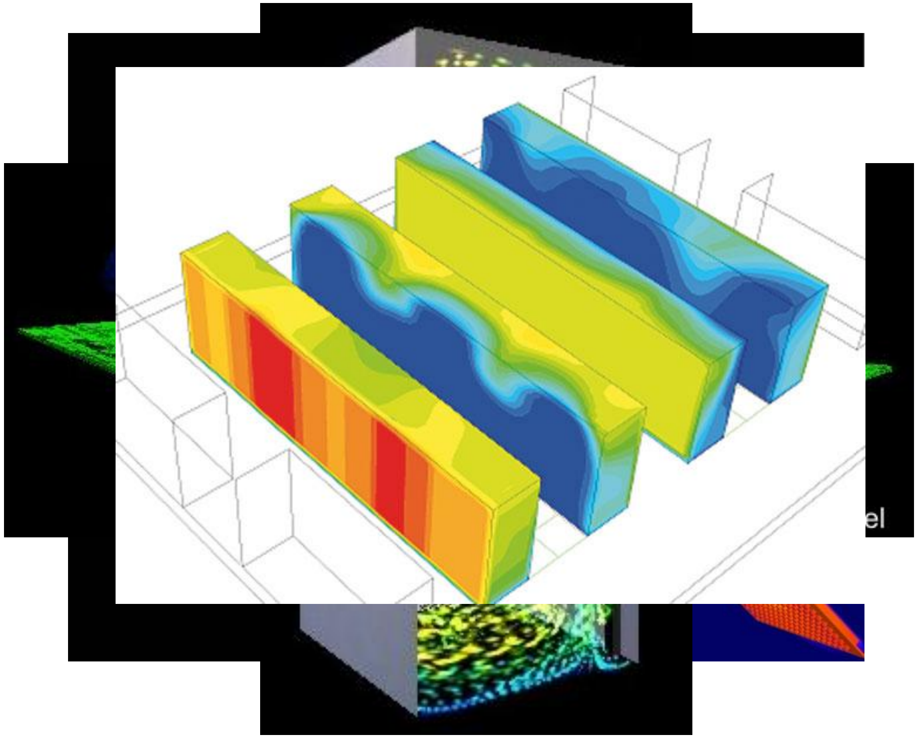


[Uptime Institute]

Cooling and hot spot avoidance is an open issue!

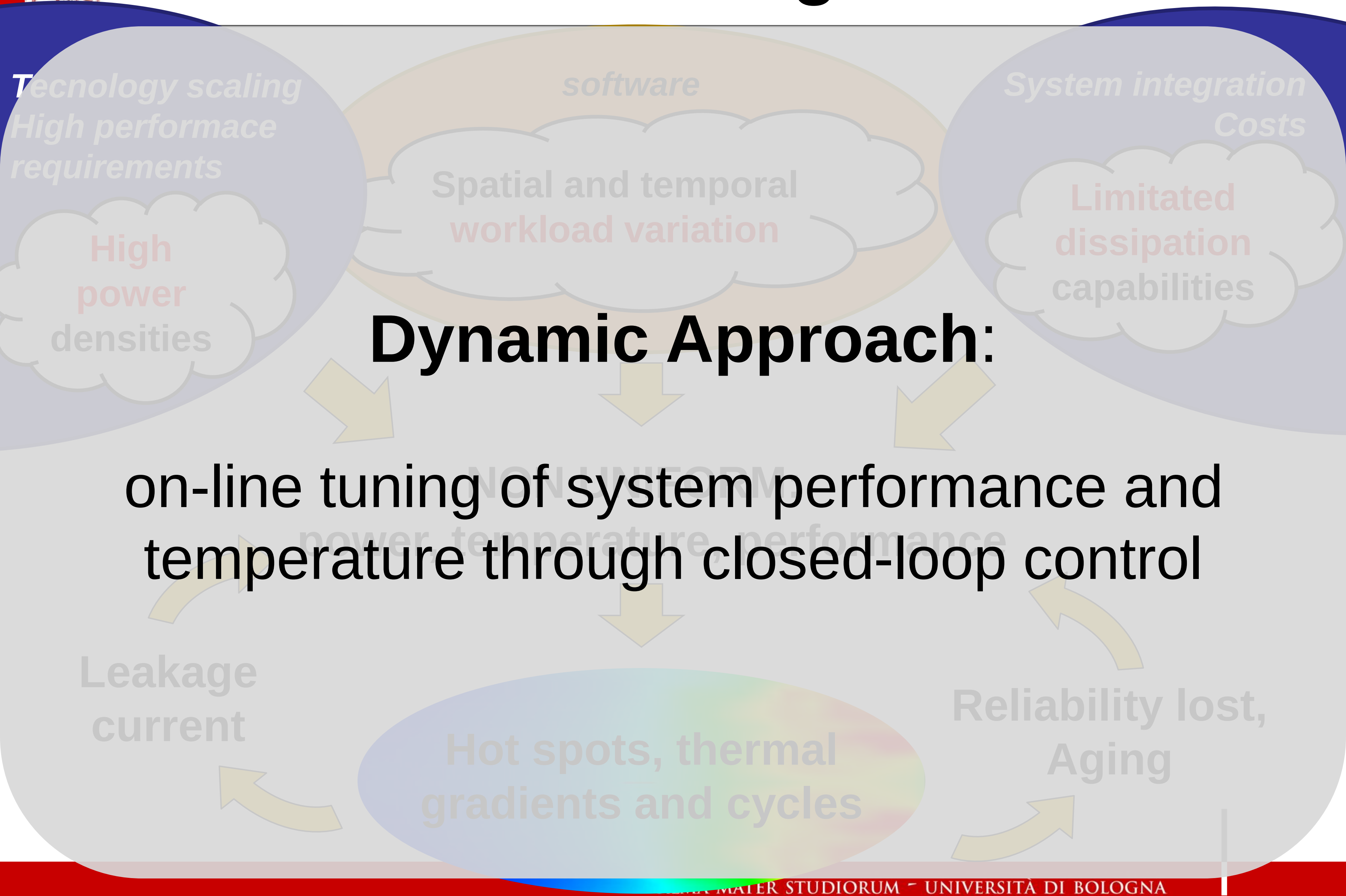
Multi-scale Problem

- Increasing power density
- Thermal issues at multiple levels
  - Chip / component level
  - Server/board level
  - Rack level
  - Room level



Today's focus: Chip level

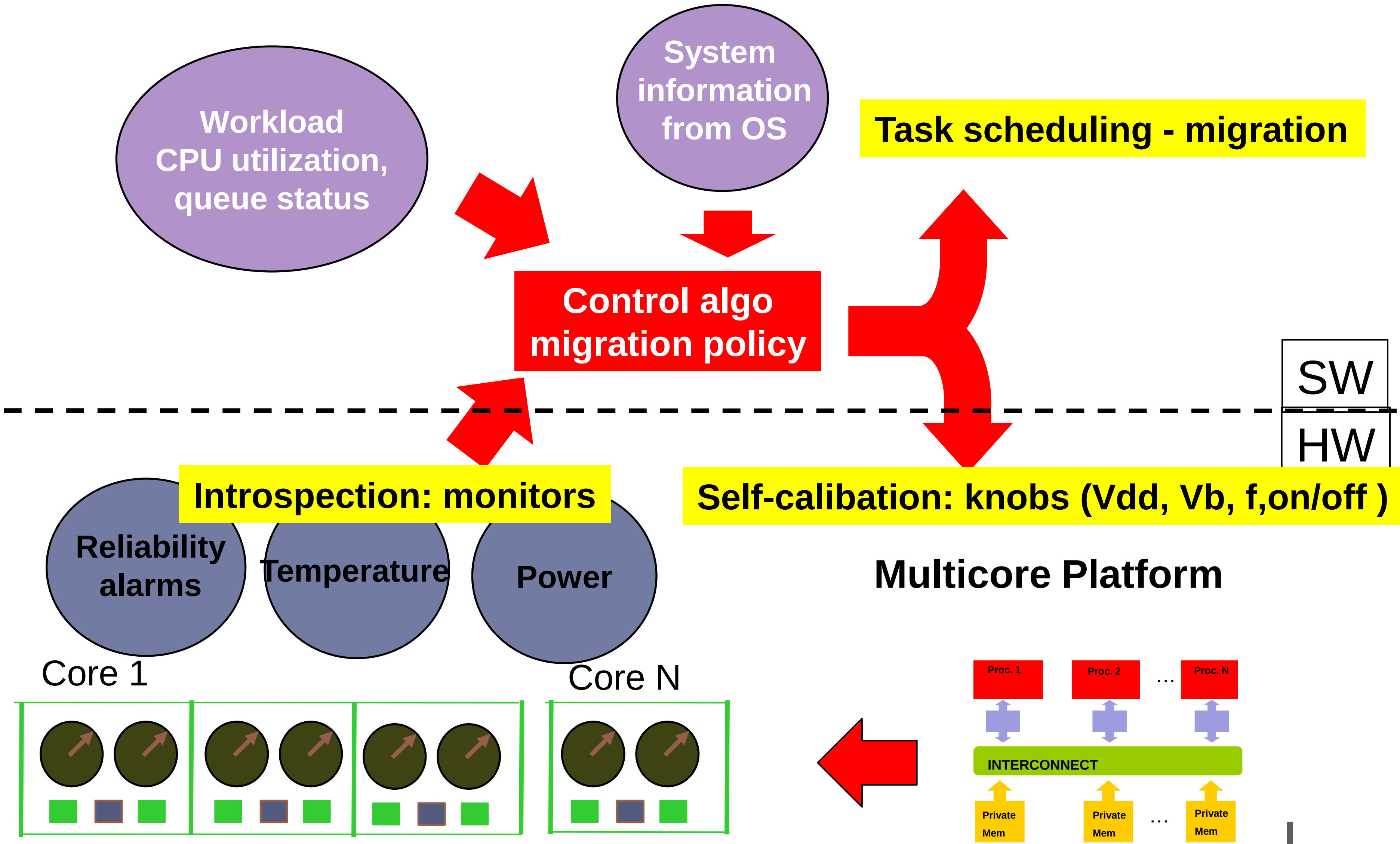
Thermal Management



Dynamic Approach:

on-line tuning of system performance and temperature through closed-loop control

Management Loop: Holistic view

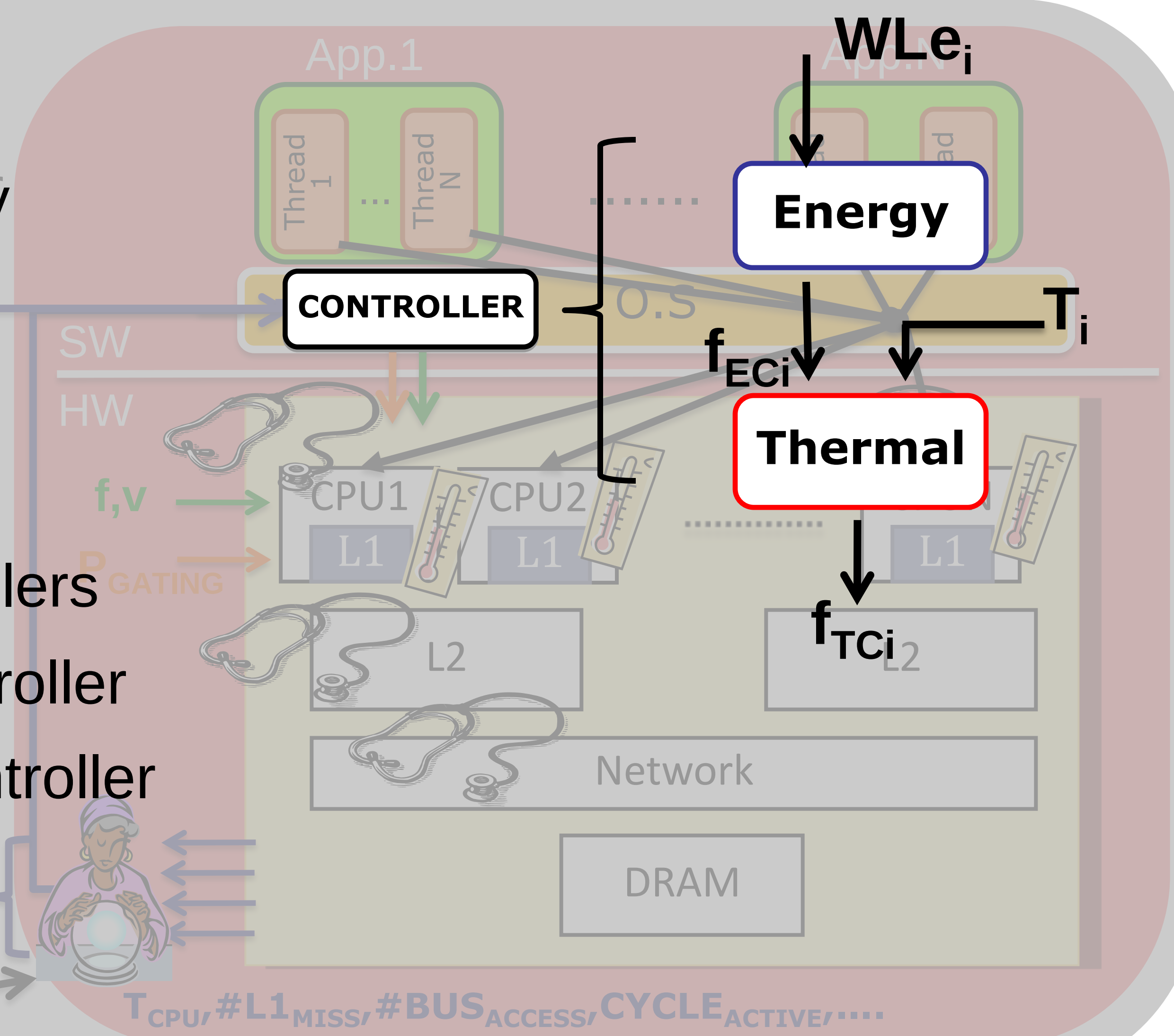


Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

DRM - General Architecture

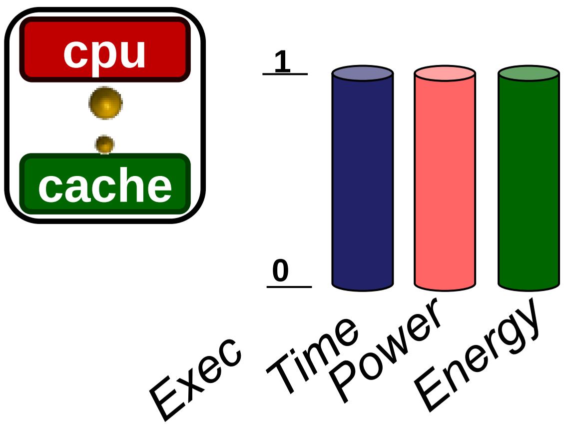
- Controller
  - Minimize energy
  - Bound the CPU temperature
- Our approach
  - Stack of controllers
    - Energy controller
    - Thermal controller



Energy Controller

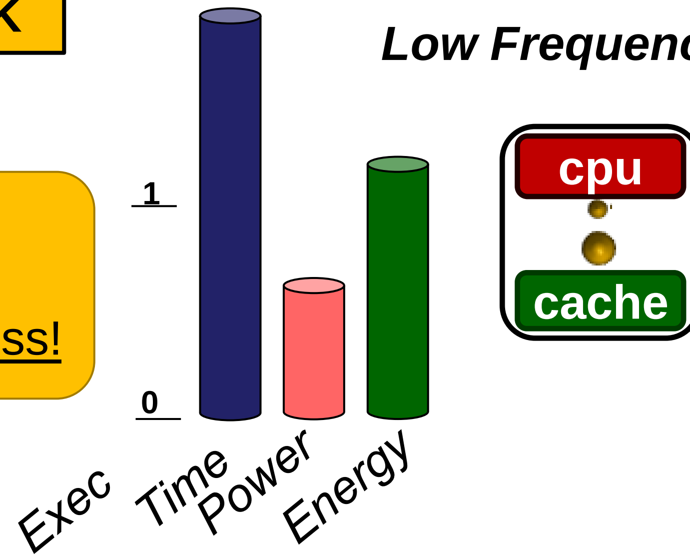
CPU BOUND TASK

High Frequency

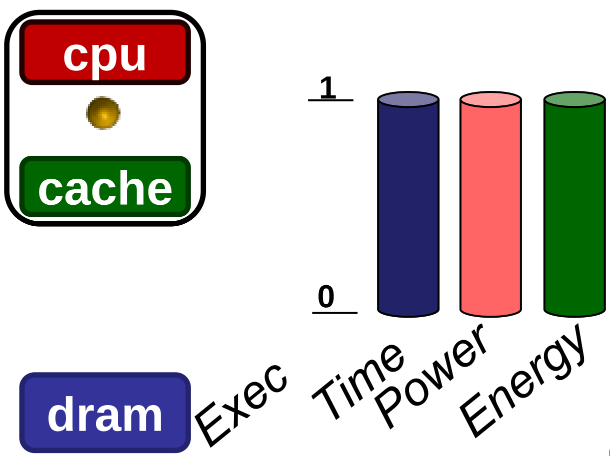


- Performance Loss
- Power reduction
- <u>Energy Efficiency Loss!</u>

Low Frequency

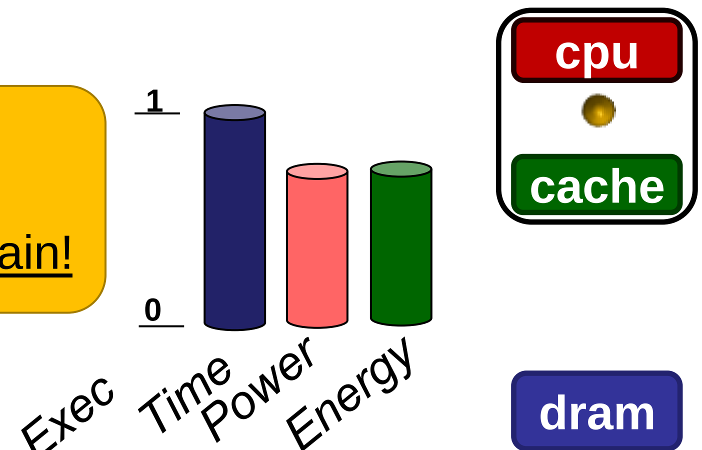


High Frequency



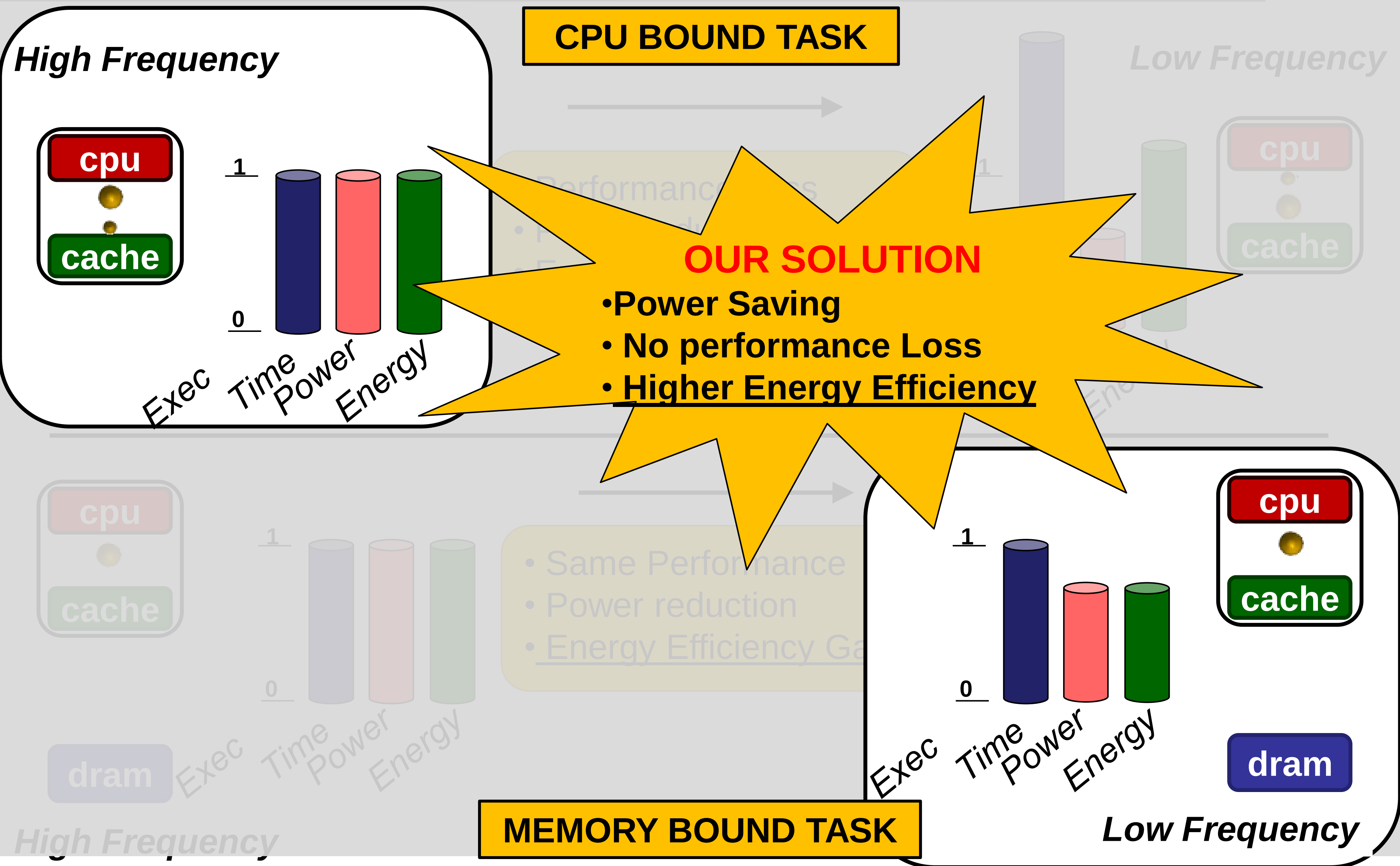
- Same Performance
- Power reduction
- <u>Energy Efficiency Gain!</u>

Low Frequency



MEMORY BOUND TASK

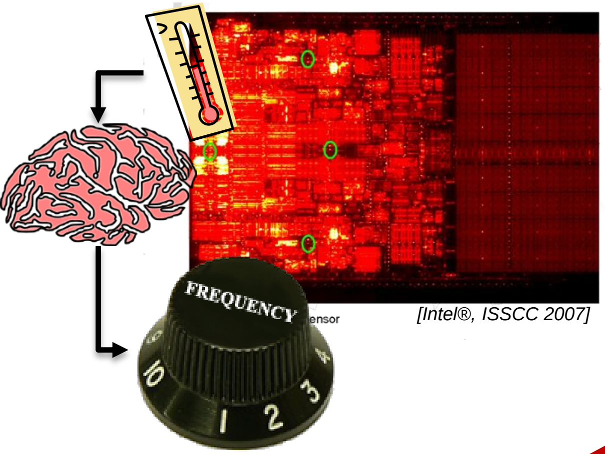
Energy Controller



Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

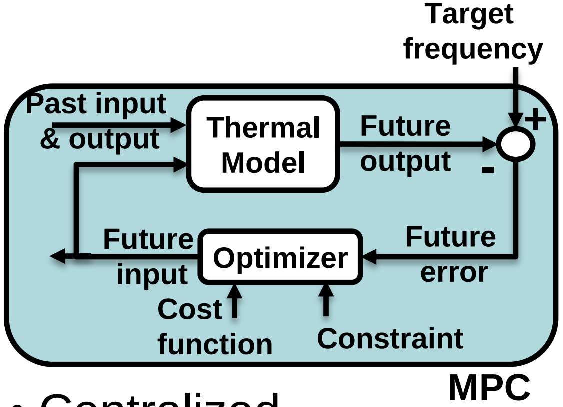
Thermal Controller



COMPLEXITY

Model Predictive Controller

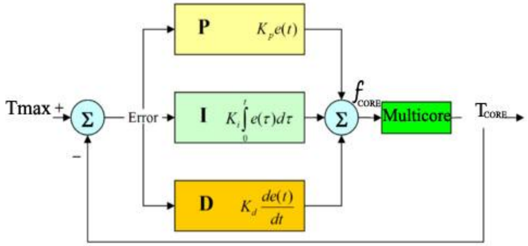
- Internal prediction:
  avoid overshoot
- Optimization:
  maximizes performance



- Centralized
  - aware of neighbor cores thermal influence
  - All at once – MIMO controller
  - Complexity !!!

Classical feed-back controller

- PID controllers
- Better than threshold based approach
- Cannot prevent overshoot

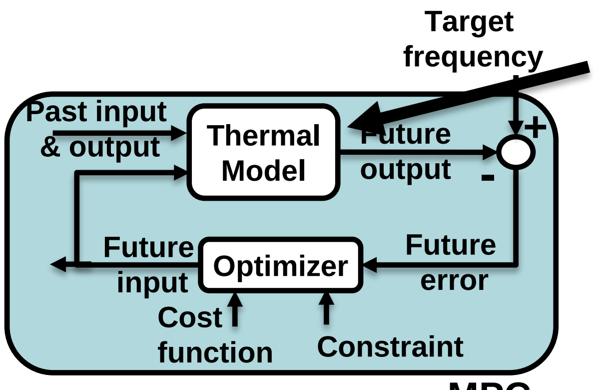


Threshold based controller

- T > Tmax → low freq
- T < Tmin → high freq
- cannot prevent overshoot
- thermal cycle

MPC Robustness

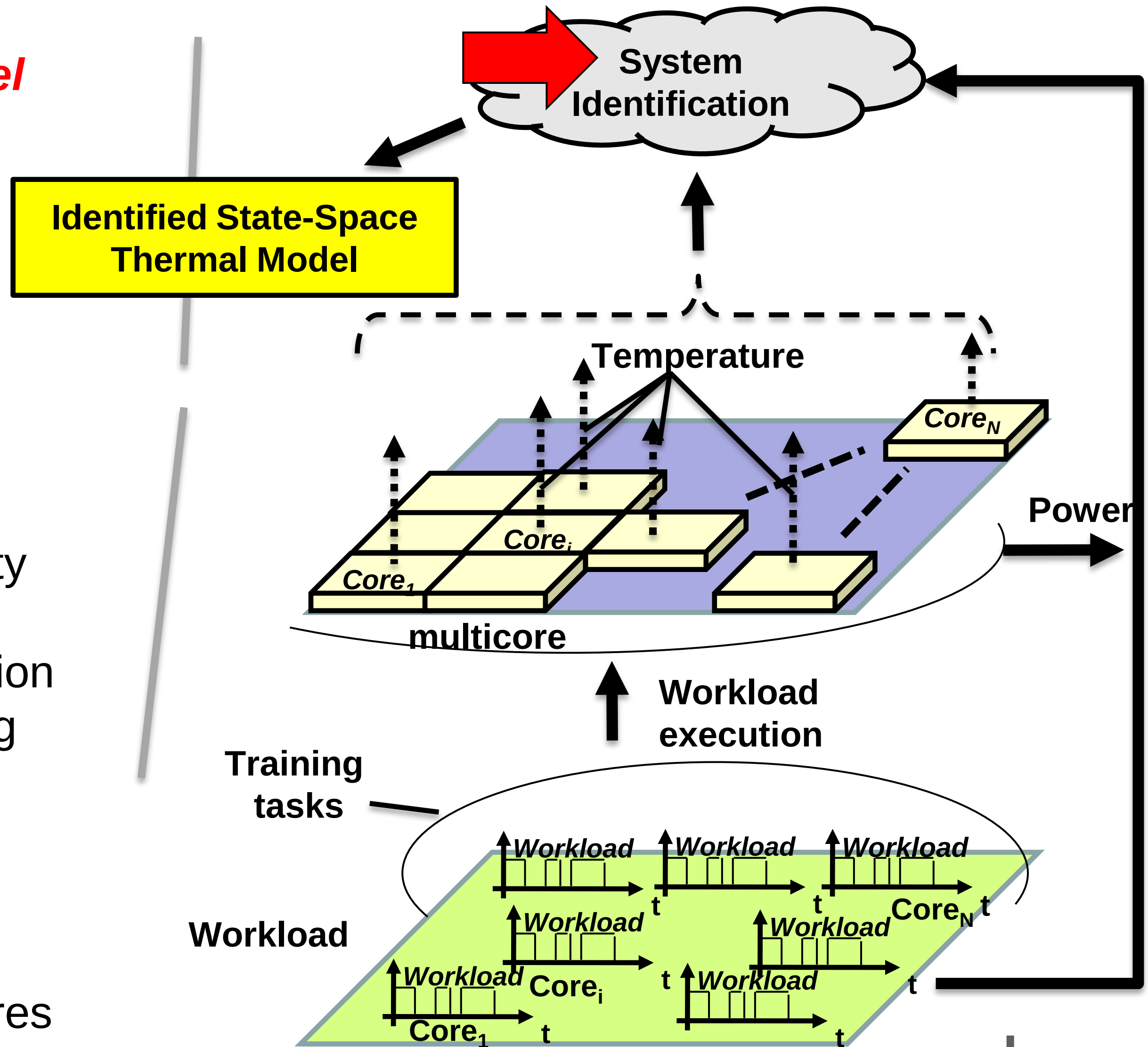
MPC needs a *Thermal Model*



- Accurate, with low complexity
- Must be known "at priori"
- Depends on user configuration
- Changes with system ageing

"In field" Self-Calibration

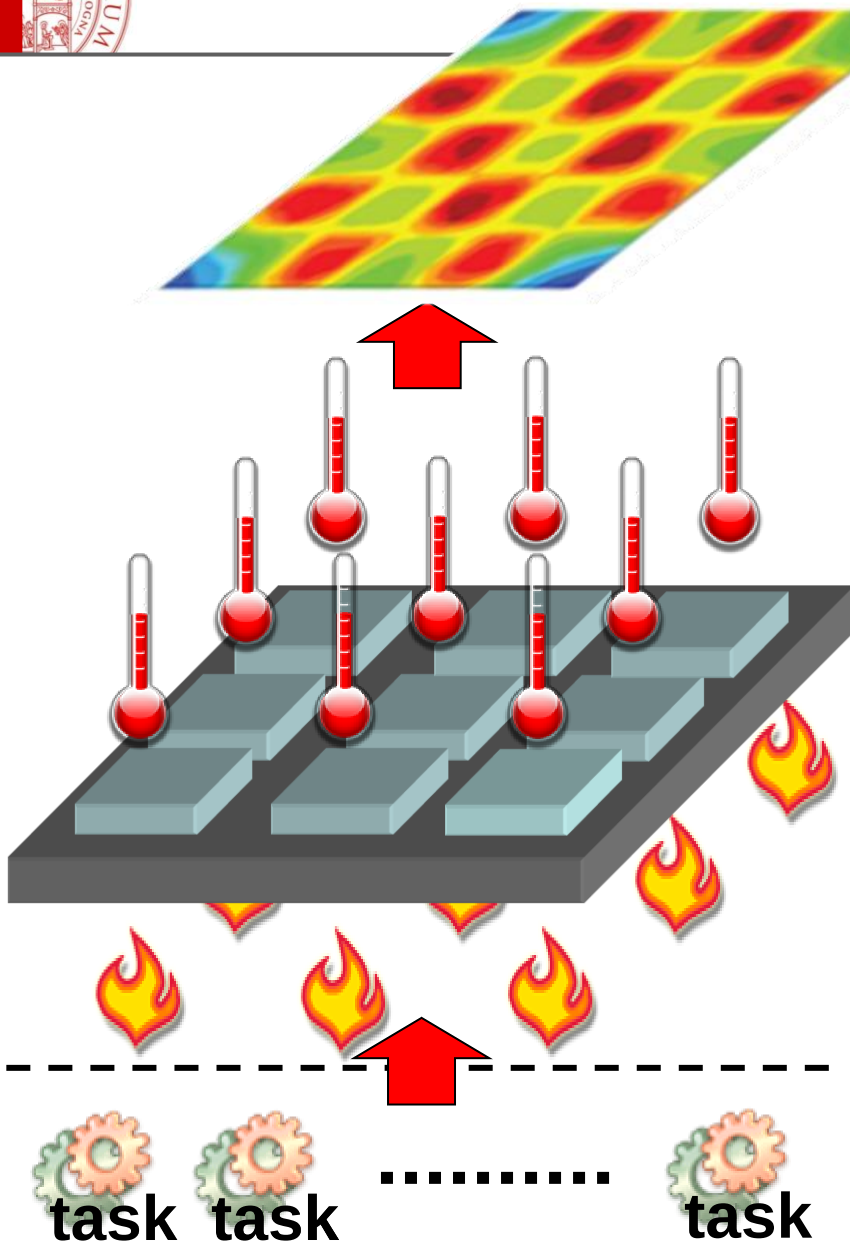
- Force test workloads
- Measure cores temperatures
- System identification

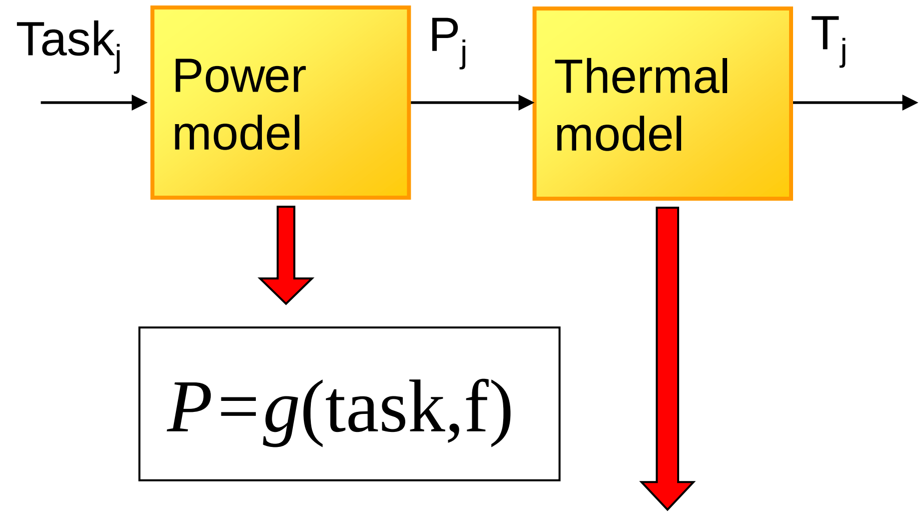


Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

Thermal Model & Power Model

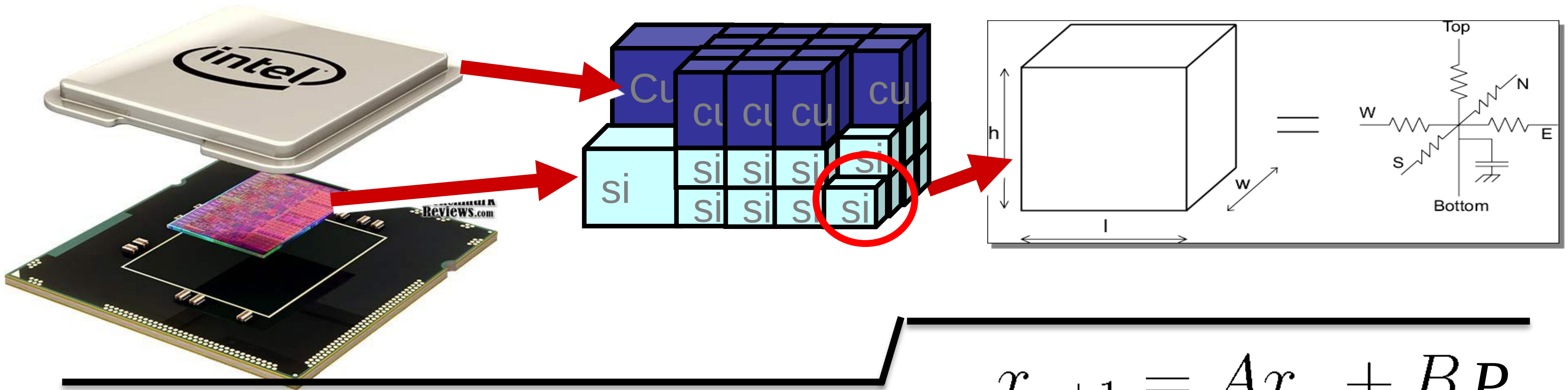




$$P = g(\text{task,f})$$

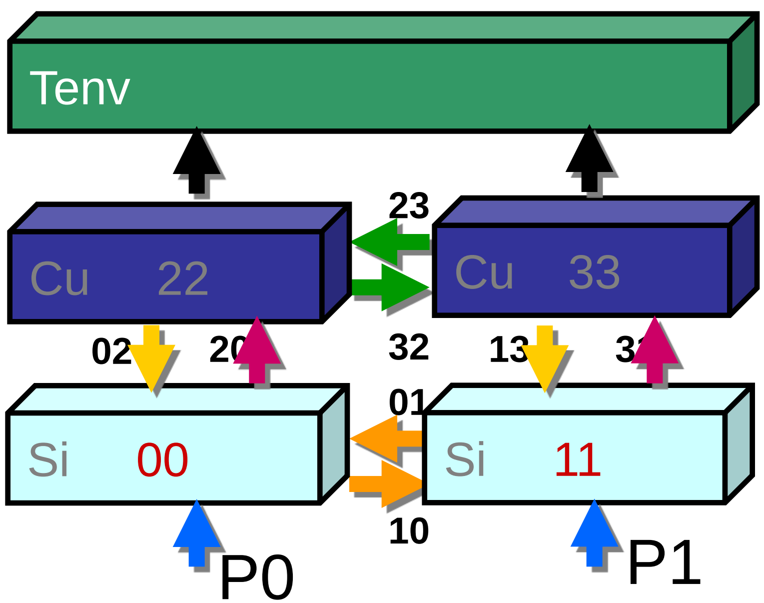
$$x_{n+1} = A x_n + B P_{n,j}$$
$$T_{n,j} = C x_n$$

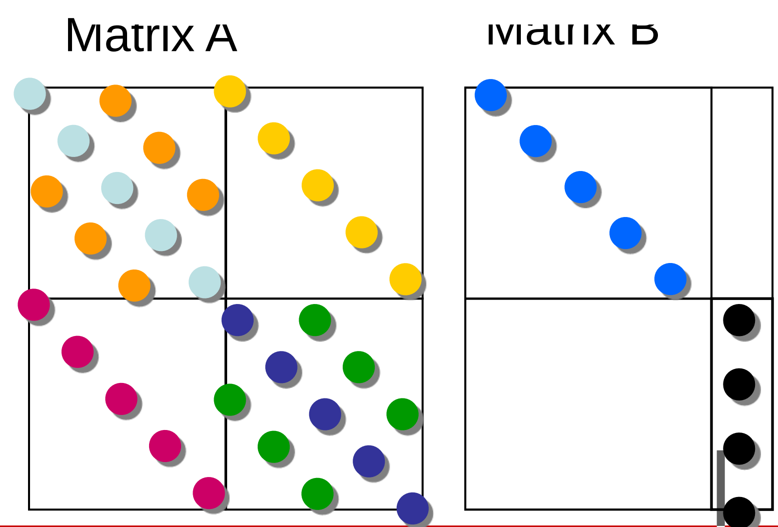
Model Structure



$$x_{n+1} = A x_n + B\, P_{n,j}$$

$$T_{n,j} = C x_n$$





LS System Identification

Test pattern

System response

Data → **Model least square optimization** → **Optimal parameters**

A,B

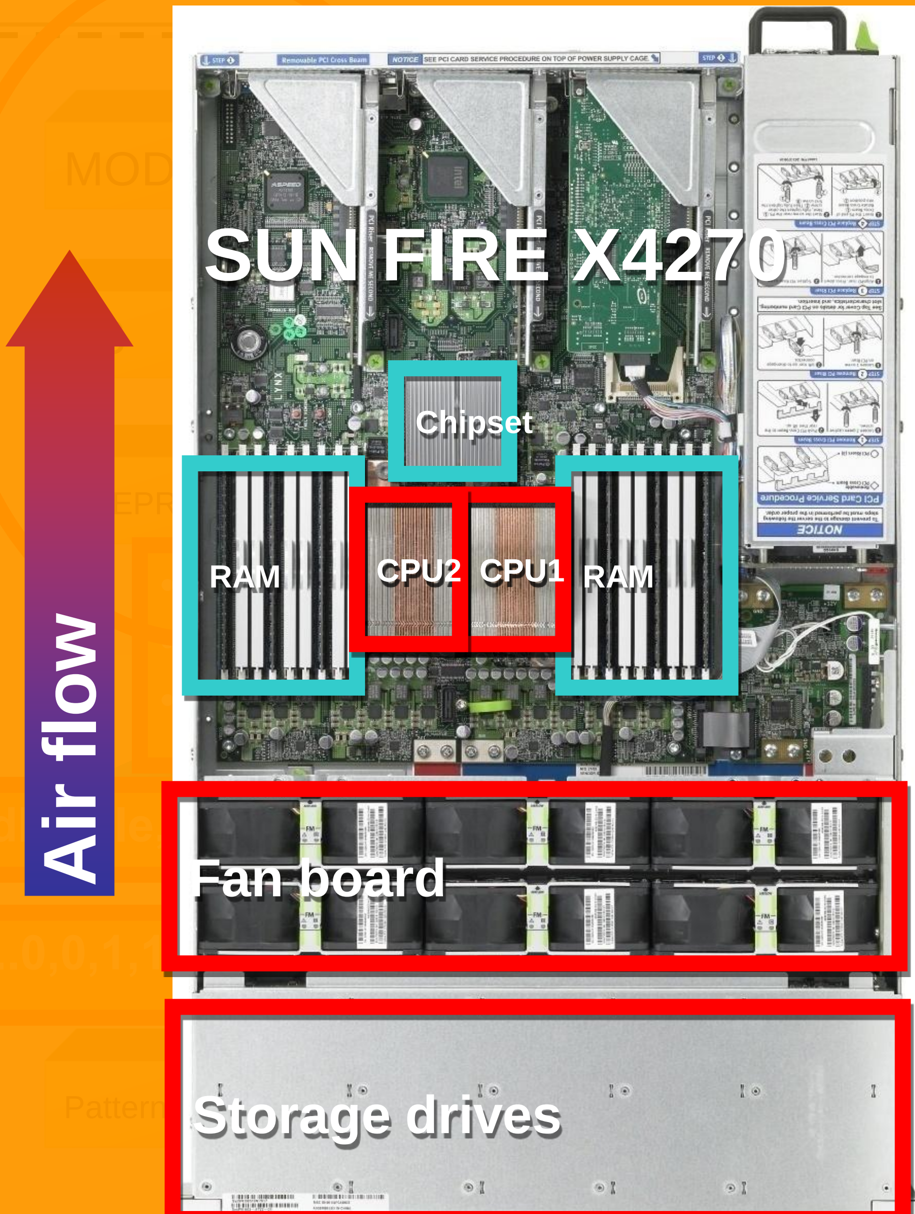
$$\theta_{opt} \rightarrow \min_{\theta} I(\theta)$$ Parametric optimization

$$\theta = \{A, B\}$$ Parameters

$$I(\theta) = \frac{1}{N_s}\sum_{i=1}^{N_s} e_i^2$$ Cost function

$$e = T_{model} - T_{real}$$ Error Function

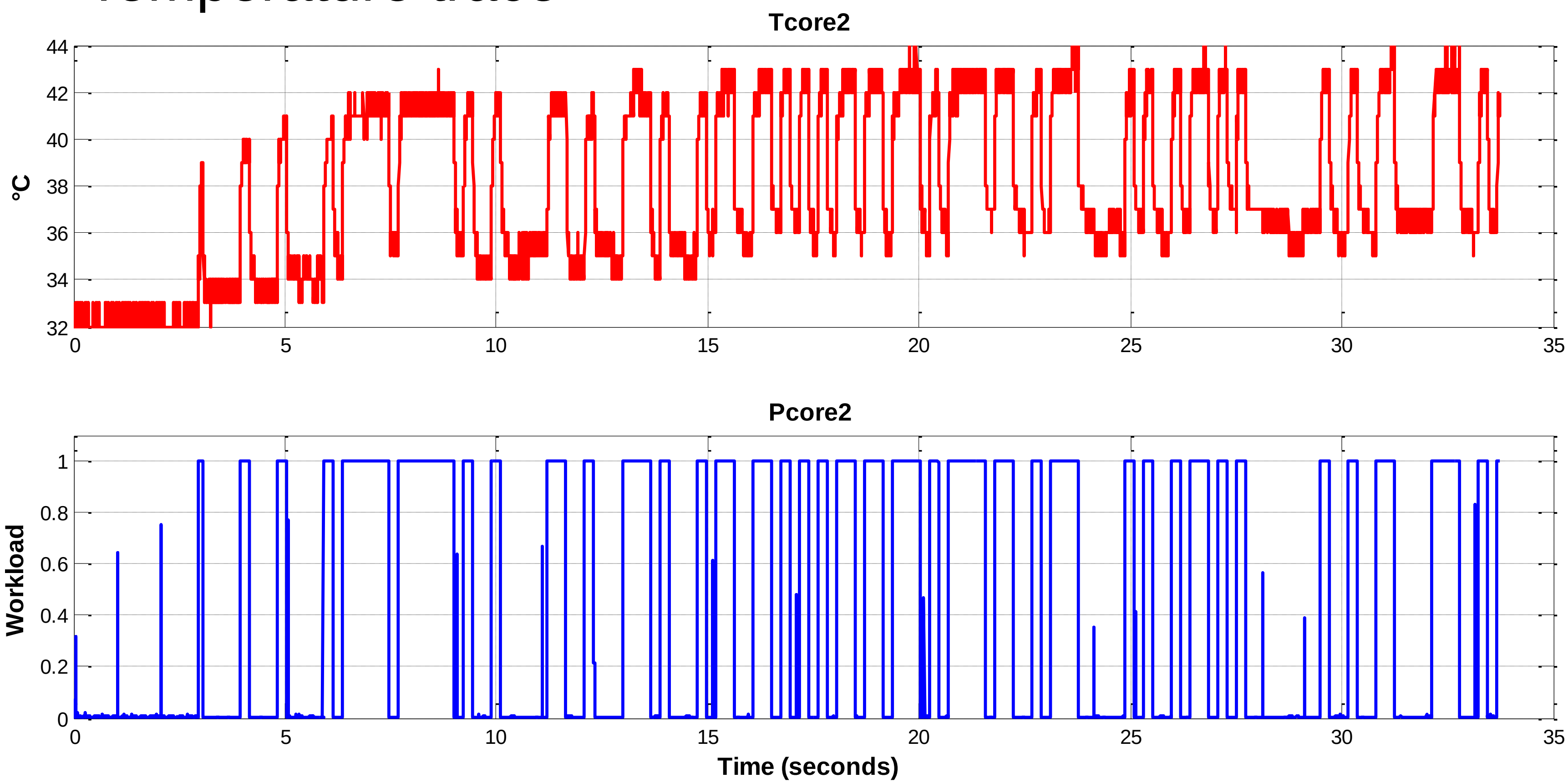
Experimental setup



- Intel Nehalem
- 8core/16thread
- 2.9GHz
- 95W TDP
- IPMI

Workload & Temperature

Temperature trace

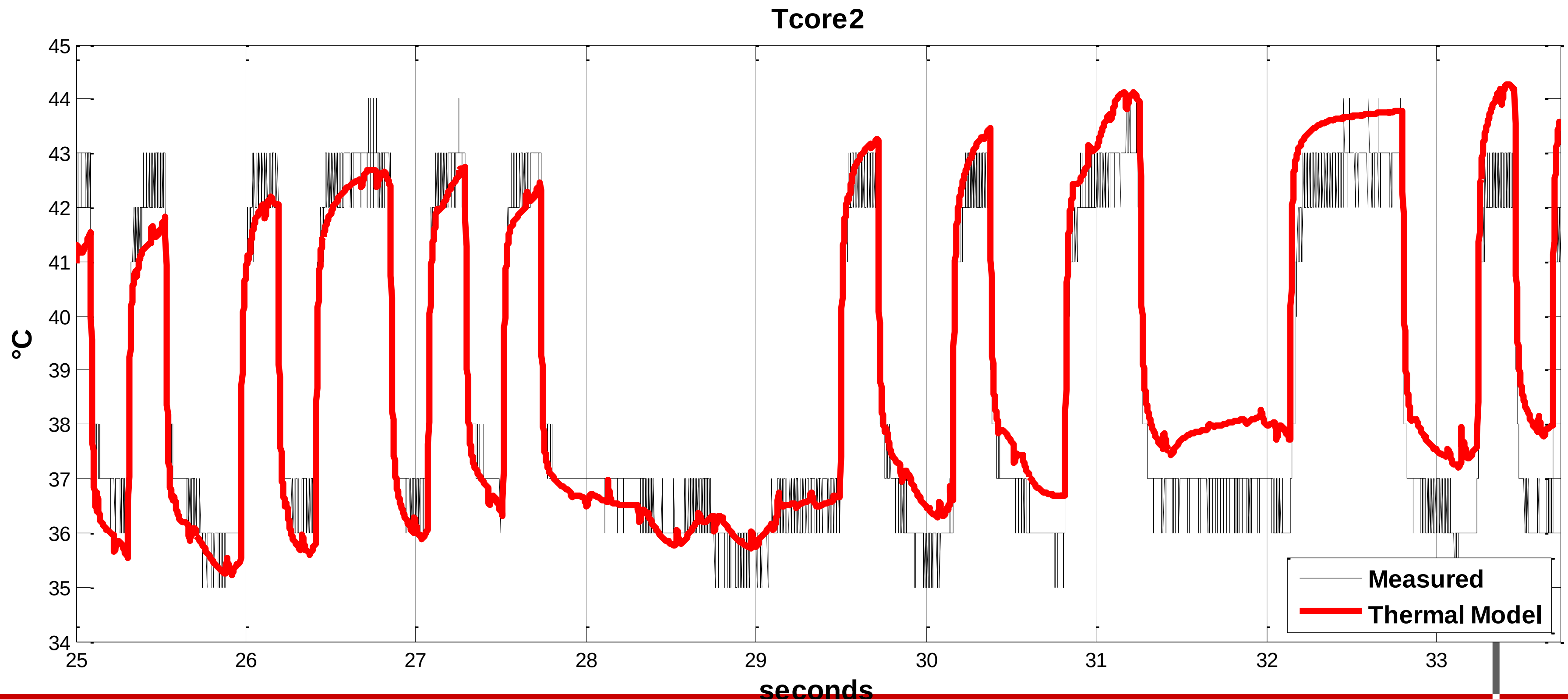


Pseudorandom workload pattern

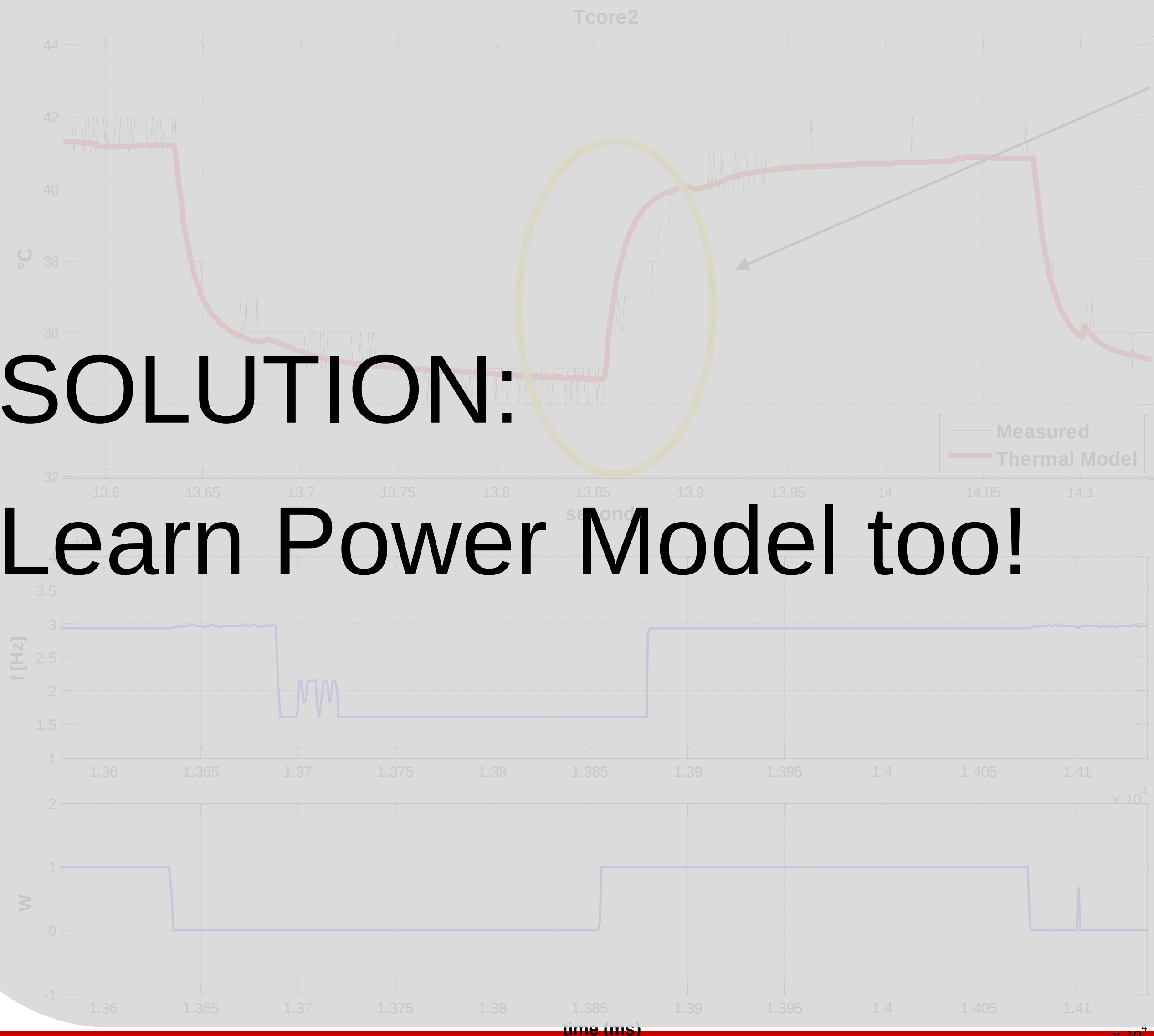
Black-box Identification

Identification based on pure LS fitting

MEASURED vs. SIMULATED TEMPERATURE



Partially unobservable model



PROBLEM 1: "WORKLOAD" signal does not include power variation due to frequency changes.

FREQUENCY

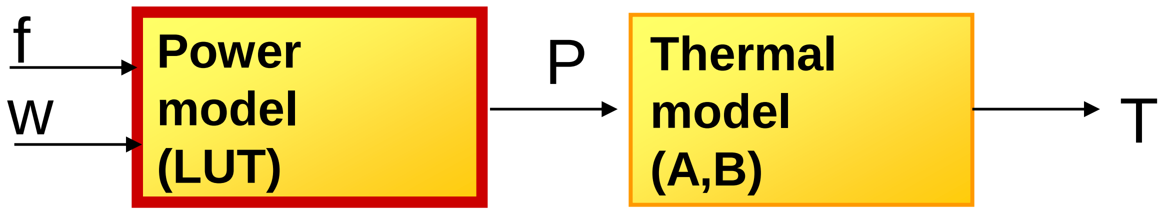
WORKLOAD

SOLUTION:

Learn Power Model too!

Multi-step Identification

Power model *P=g(w,f) initially unknown*



1° STEP: set *f=const*, set *w as* $[0|1]^N$ *sequence* → $[P1|P0]^N$ *with P1, P0 pre-measured in steady state*, we measure *T* to obtain A_0 by LS

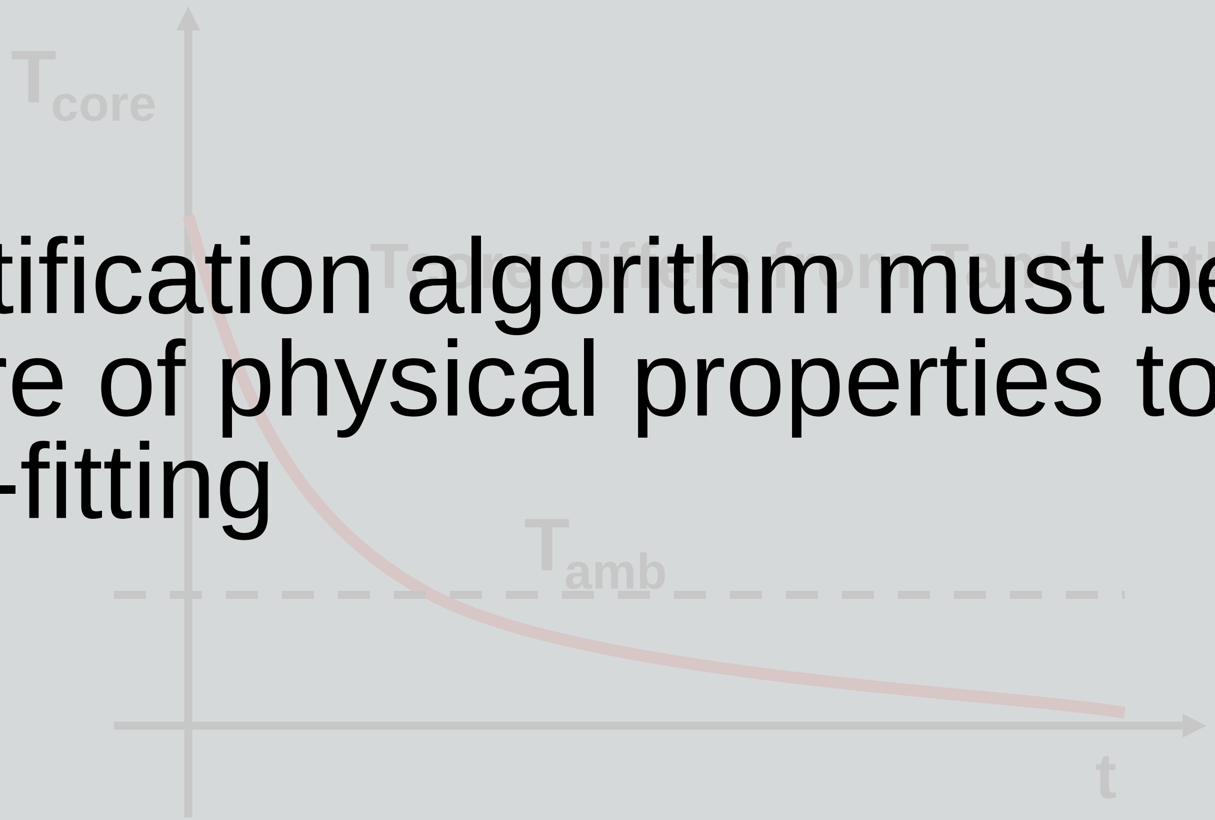
2° STEP: A is known, we set f, w, we measure T, we invert A and we obtain P

3° STEP: P is known, we now generate richer sequence w,f and we re-calibrate A by LS

Iterate until convergence

Validation

Problem 3: Model is not physical

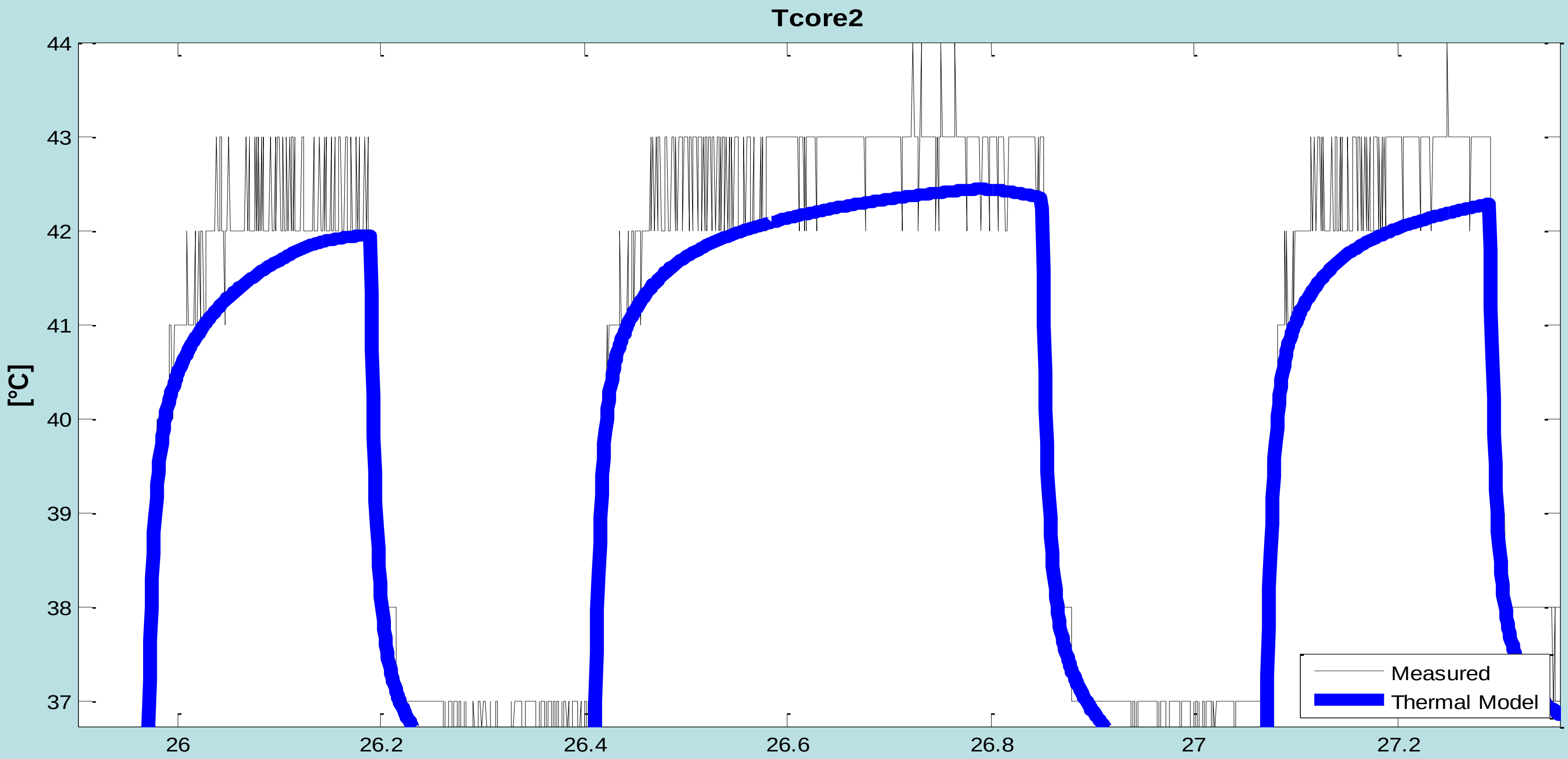


Identification algorithm must be aware of physical properties to avoid over-fitting

time (ms)

x 10^4

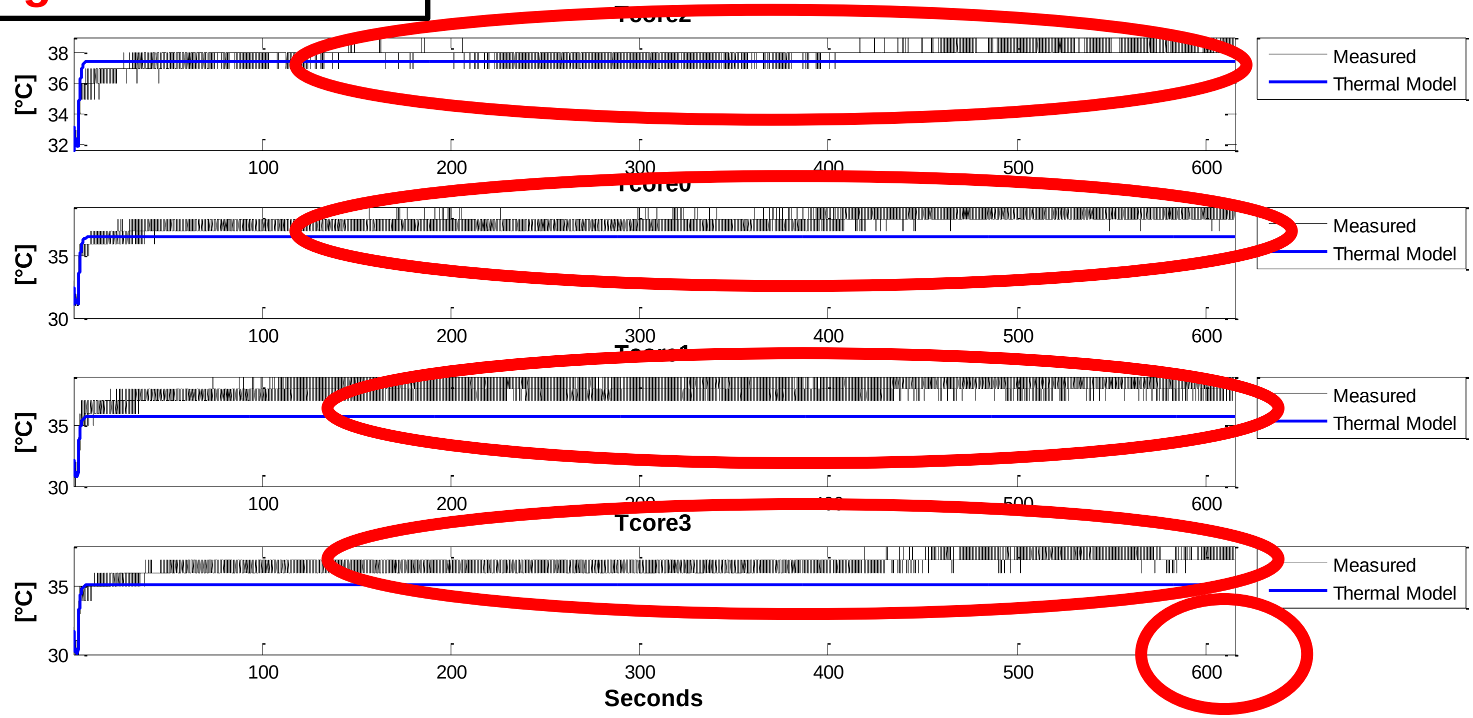
Constrained Identification



0.8393	0.9699	0.9956	0.9998	0.9418	0.9053	0.9029	0.9186

Quasi-steady-state accuracy

Large time constant



Possible causes:

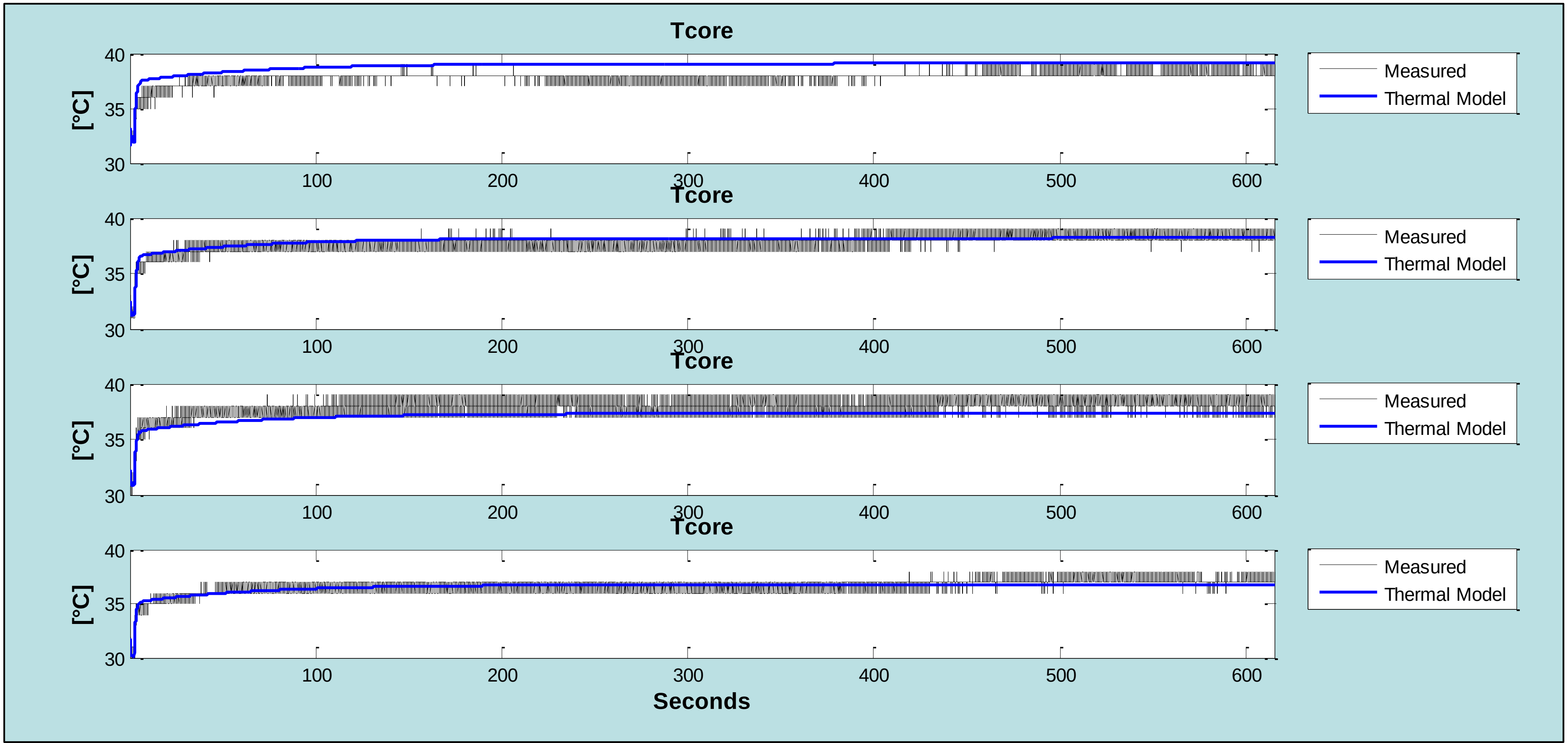
- Package thermal inertia?
- Environment inertia (Air)?
- P_{LEAK} temperature dependency?

Identification with pseudorandom trace:

- Too many samples, huge LS computation

Addressing models stiffnes

- **Modelling the third time constant as heat sink temperature variation**
- **One-pole model identification**

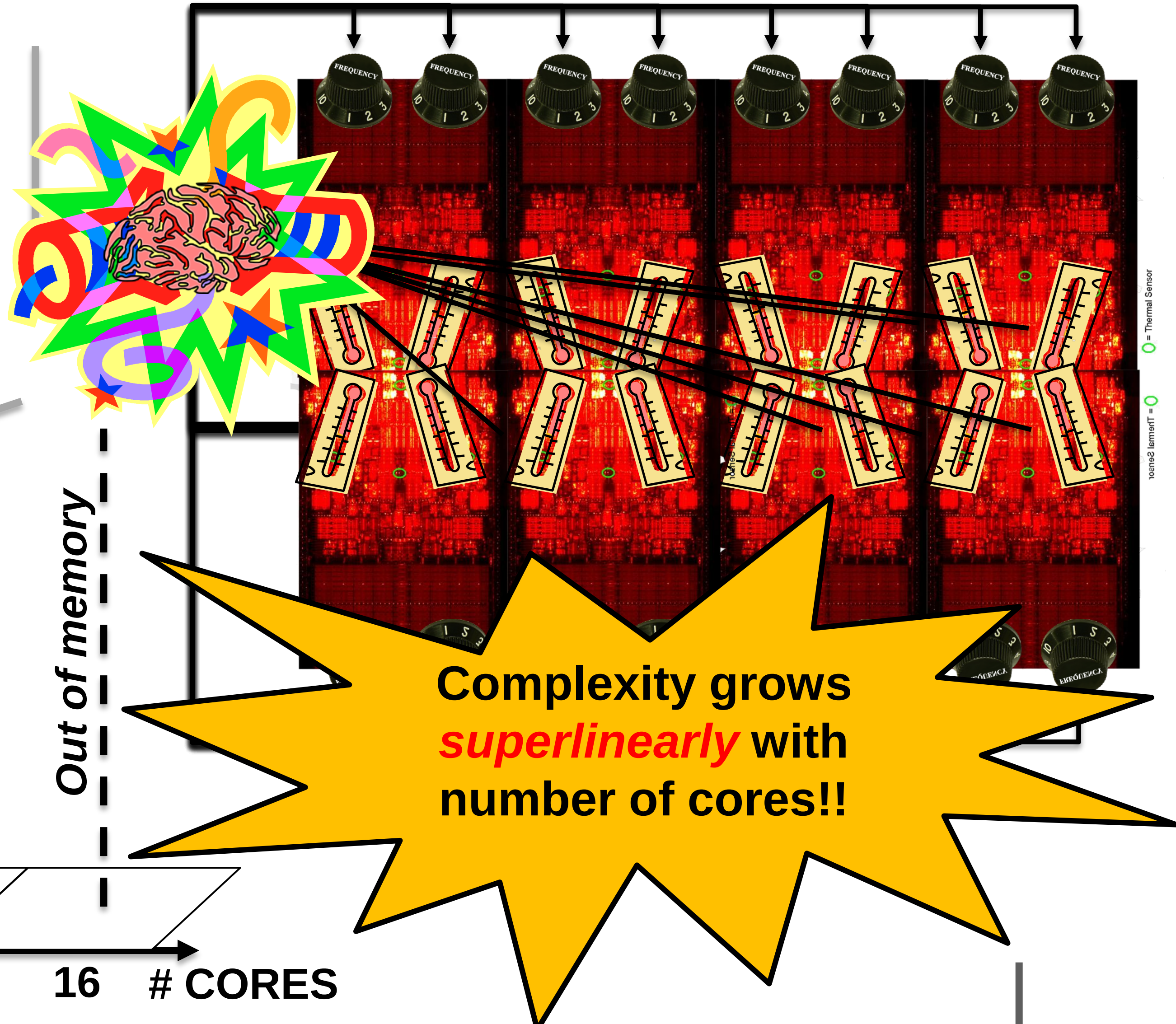


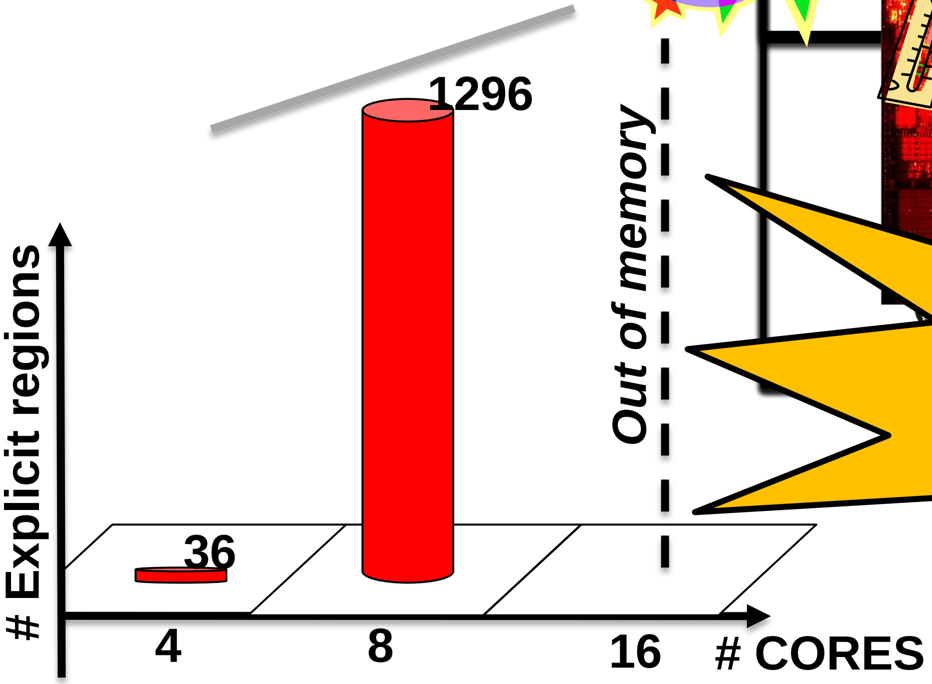
Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

MPC Scalability

MPC ***Complexity***
- **Implicit** - *a.k.a. on-line*
  - computational burden
- **Explicit** – *a.k.a. off-line*
  - high memory occupation

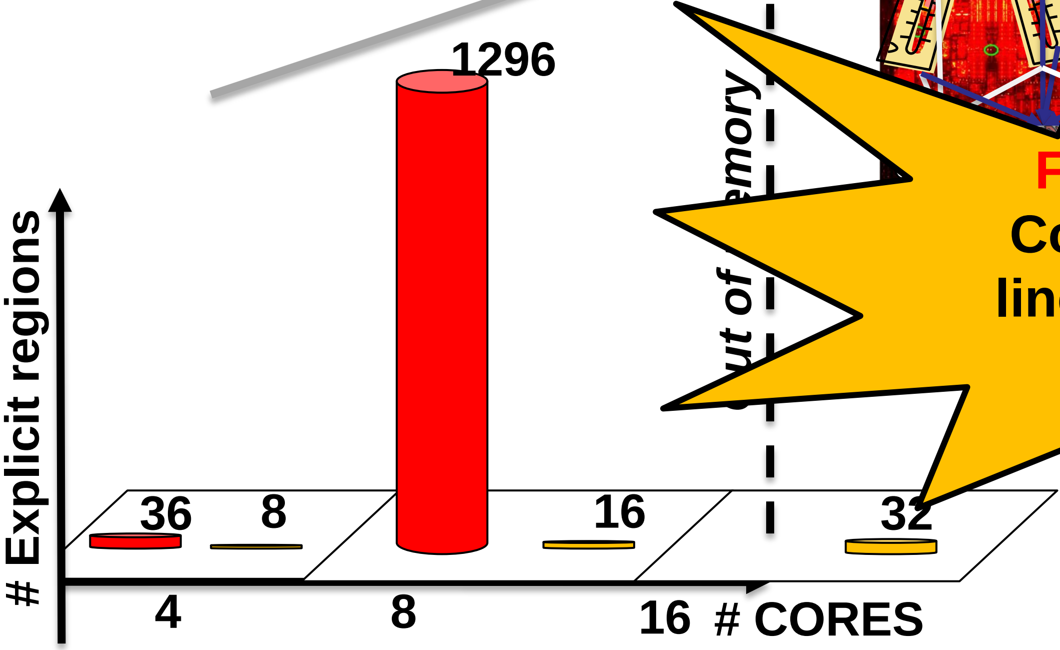




Addressing Scalability

On a short time window, power has a local thermal effect!

One controller for each core
Controller uses:
- local power & thermal model
- neighbor's temperatures

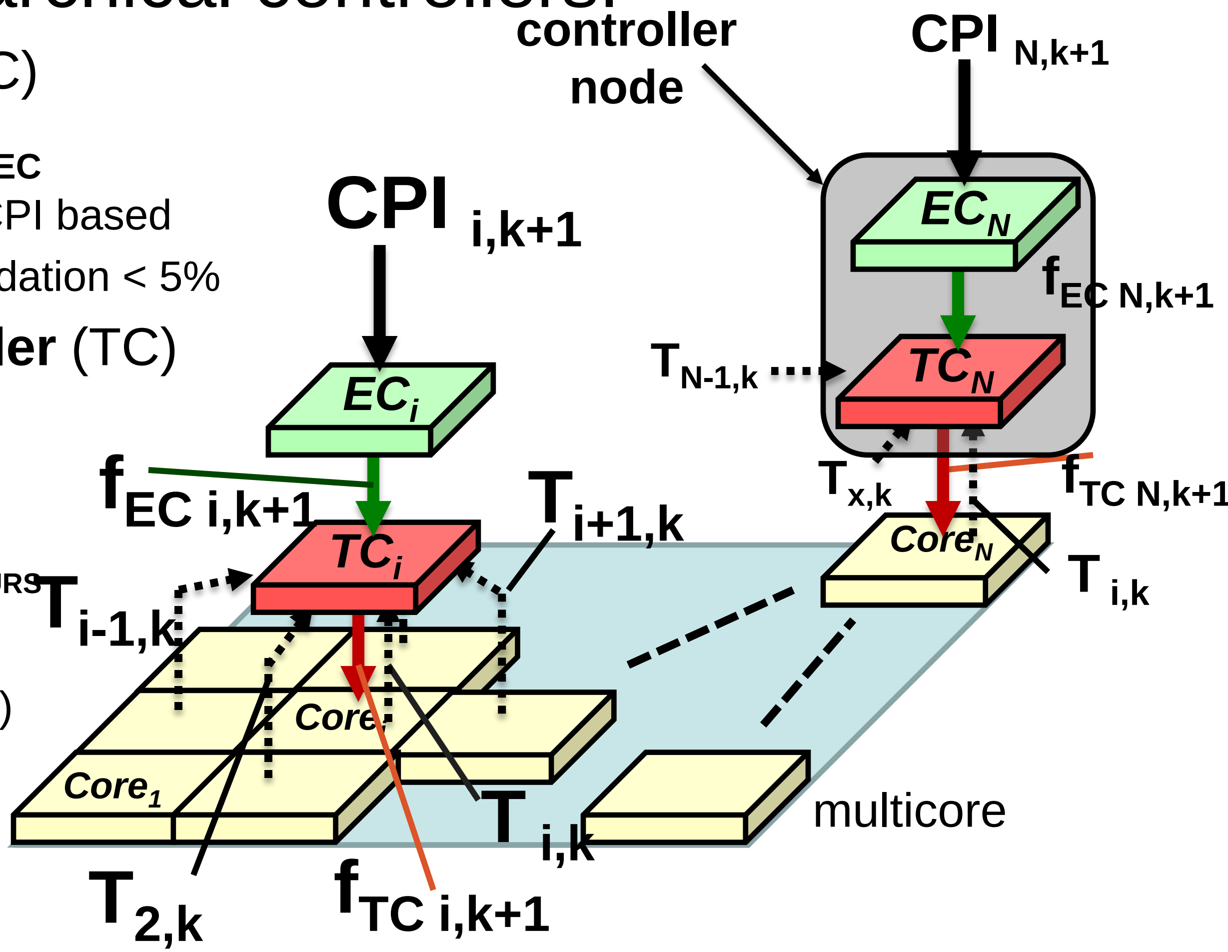


Fully distributed
Complexity scales linearly with #cores

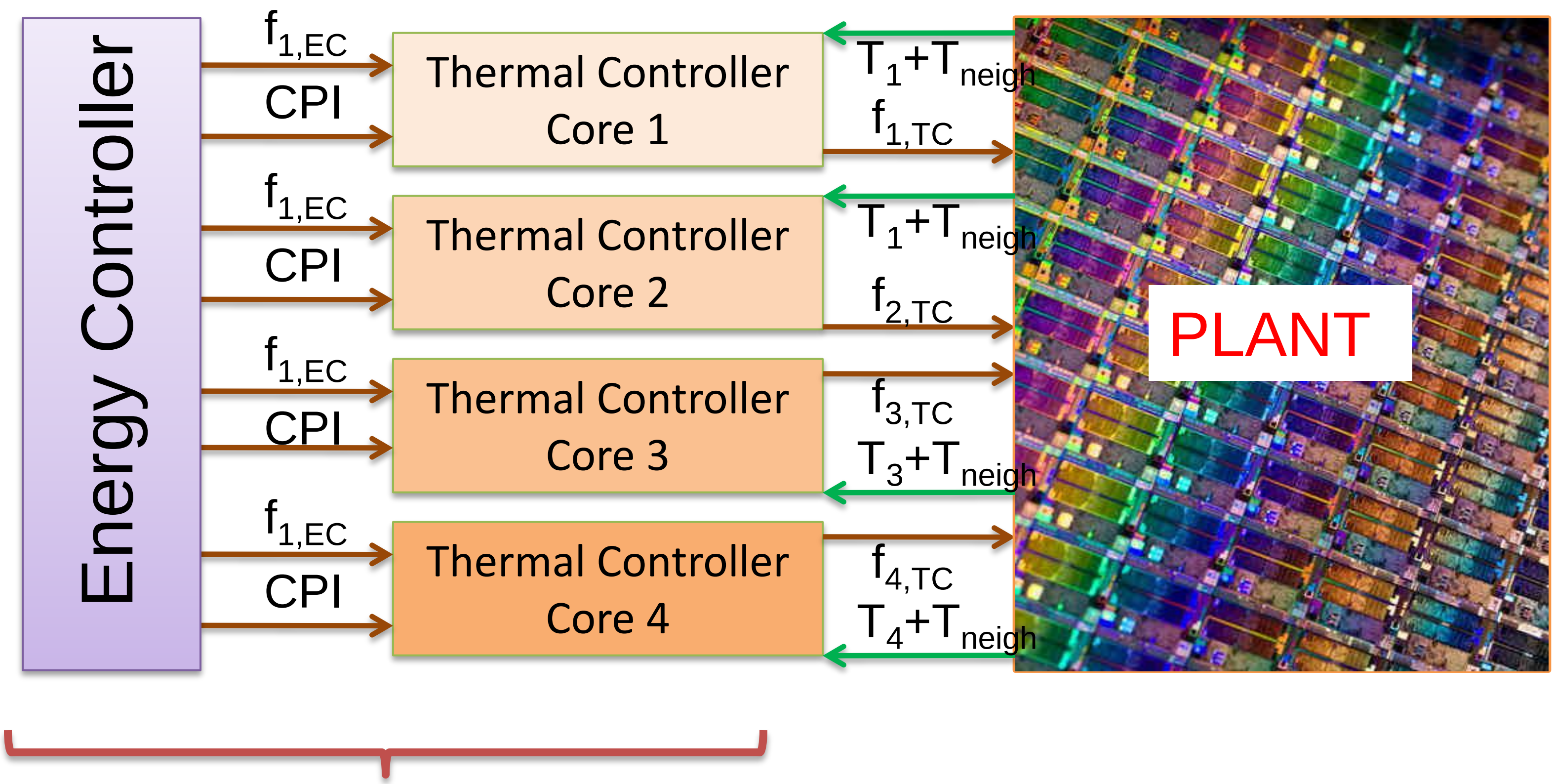
Distributed Control

Distributed and hierarchical controllers:

- **Energy Controller** (EC)
  - Output Frequency $\mathbf{f_{EC}}$
    - Minimize power – CPI based
    - Performance degradation < 5%
- **Temperature Controller** (TC)
  - Distributed MPC
  - Inputs:
    - f_{EC}, T_{CORE}, $T_{NEIGHBOURS}$
  - Output
    - Core frequency ($\mathbf{f_{TC}}$)

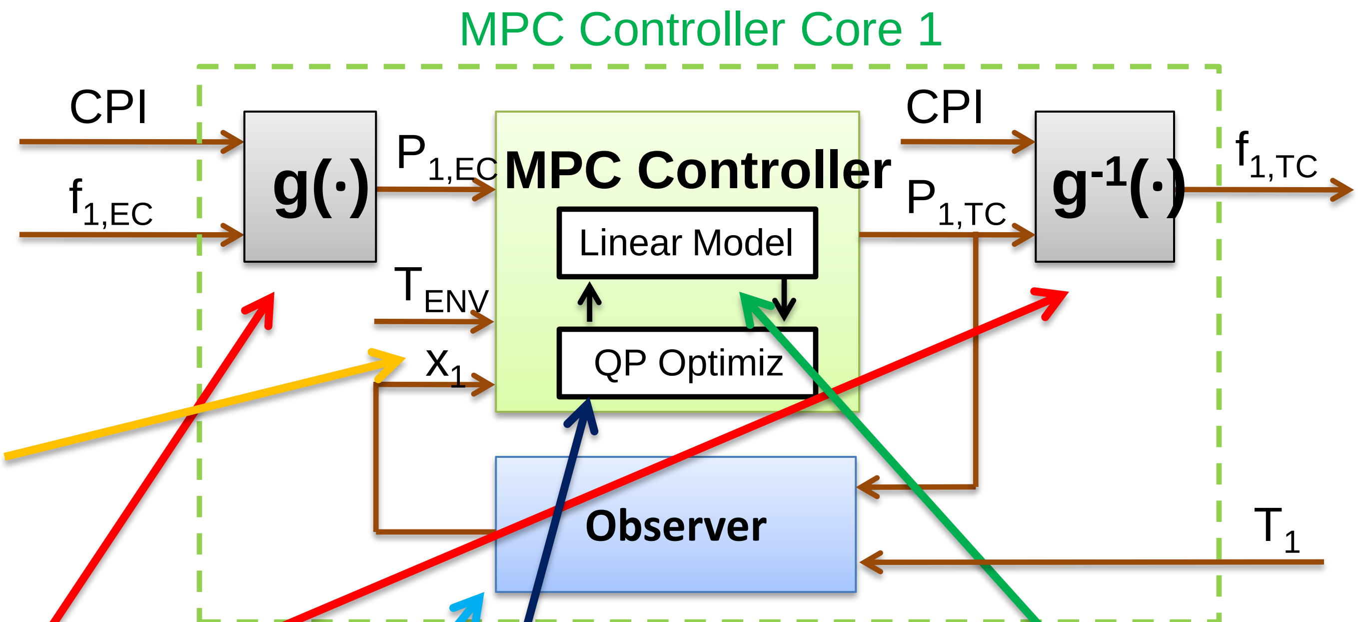


High Level Architecture



Distributed Controller

Distributed Thermal Controller



Implicit formulation

2 states per core

Nonlinear (Frequency to Power)

Linear (Power to Temperature)

$$\min \sum_{k=0}^{N-1} \| Q \cdot (f_{TC,1}(t+k) - f_{EC,1}(t+k)) \|_2$$

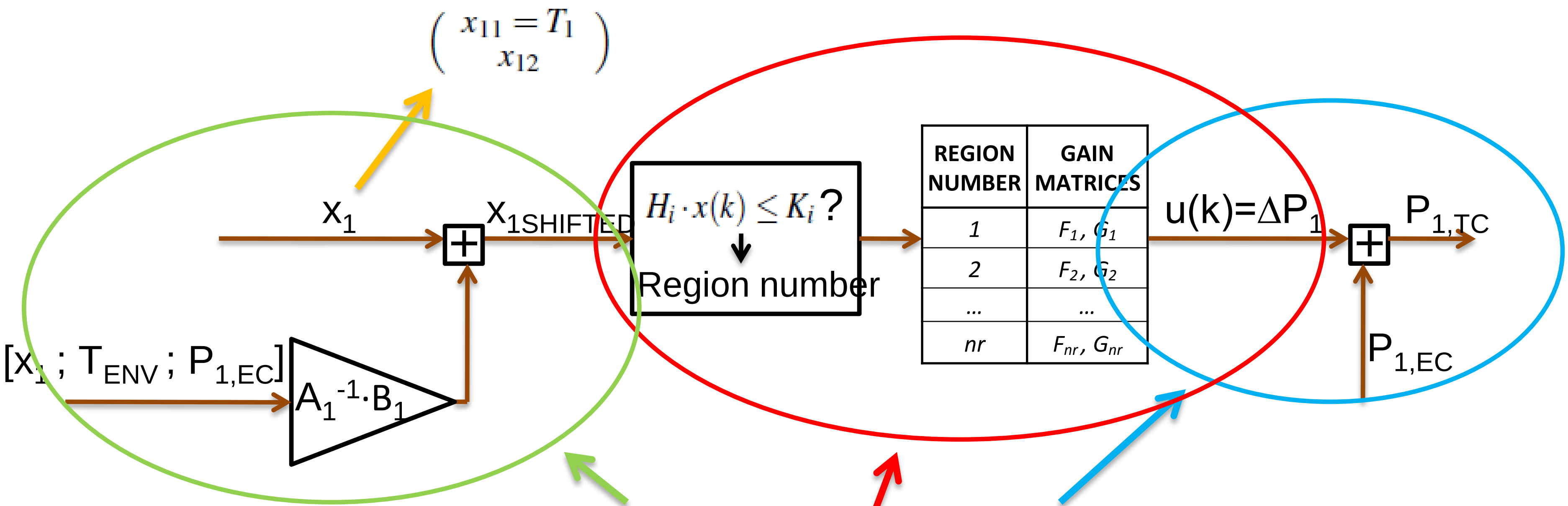
s.t

$$0 \le T_1(t+k|t) \le T_{MAX,1}$$

$$\hat{x}(k+1) = A_1 \cdot \hat{x}(k) + B_1 \cdot u(k) + L_1 \cdot (T_1 - C_1 \cdot \hat{x}(k))$$

Classic Luenberger state observer

Explicit Distributed Controller



The prediction evaluated by our explicit controller cannot take into account the measured disturbances (u_{MD}=[Tenv, P1, T_{neigh}]). Thus we exploit the superposition principle of linear systems:

$$x(k+1) = f(x(k), u_{MV}(k), u_{MD}(k)) \rightarrow x(k+1) = f(x(k), u_{MV}(k), 0) + f(x(k), 0, uMD(k))$$

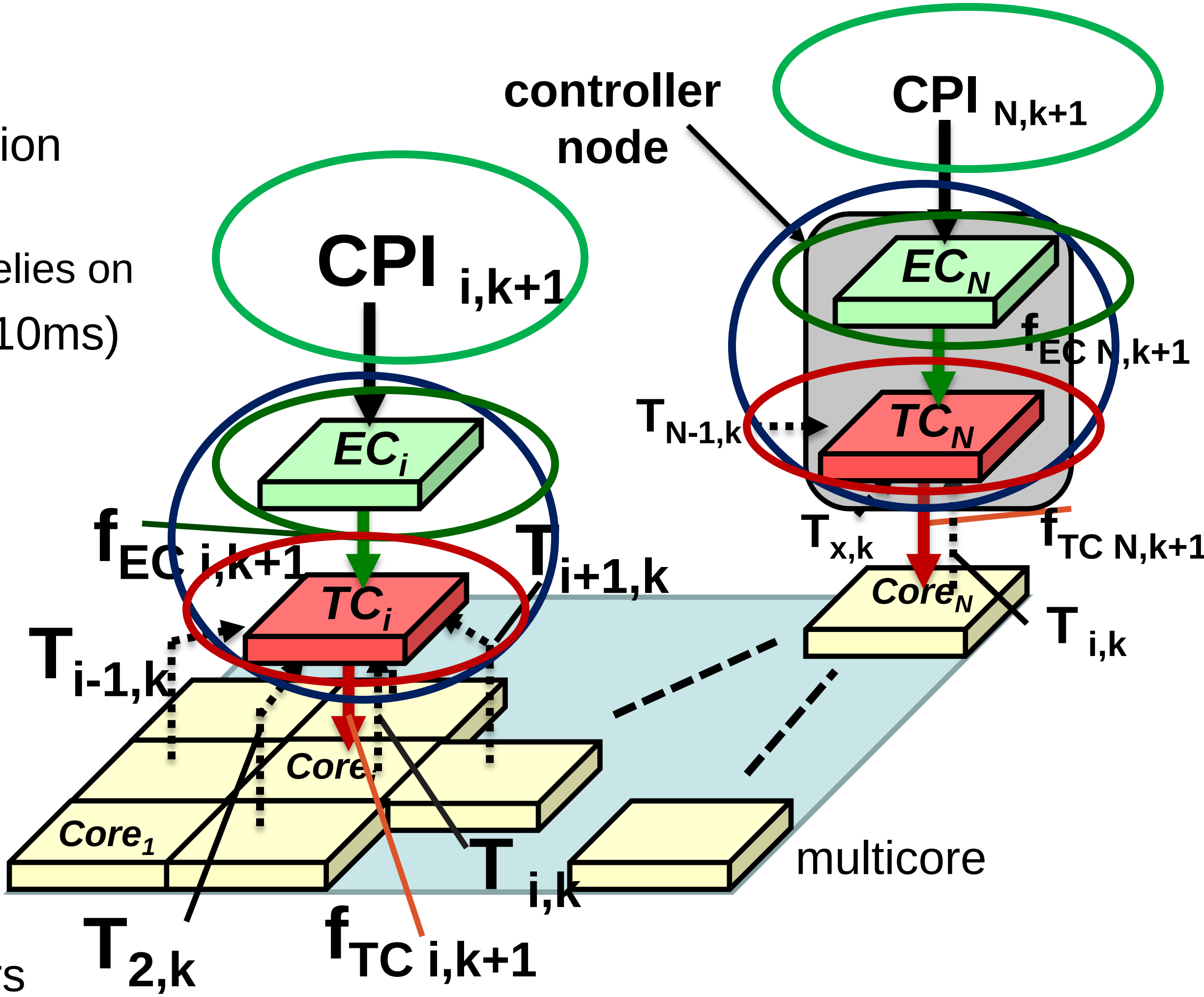
To remap the effect of these elements we exploit the model to modify the state (x(k) → $x_{SHIFTED}$(k)) projecting one step forward the MDs effects.

$$x(k+1) = A_1 \cdot x(k) + B_1' \cdot \Delta P_1 + B_1'' \cdot [P_{1,EC} \; T_{ENV} \; T_{neig}] \rightarrow x(k+1) = A_1 \cdot x_{SHIFTED}(k) + B_1' \cdot \Delta P_{1,EC}$$

$$A_1 \cdot x_{SHIFTED}(k) = A_1 \cdot x(k) + B_1'' \cdot [P_{1,EC} \; T_{ENV} \; T_{neig}] \rightarrow x_{SHIFTED}(k) = x(k) + A_1^{-1} \cdot B_1'' \cdot [P_{1,EC} \; T_{ENV} \; T_{neig}]$$

O.S.Implementation – Linux SMP

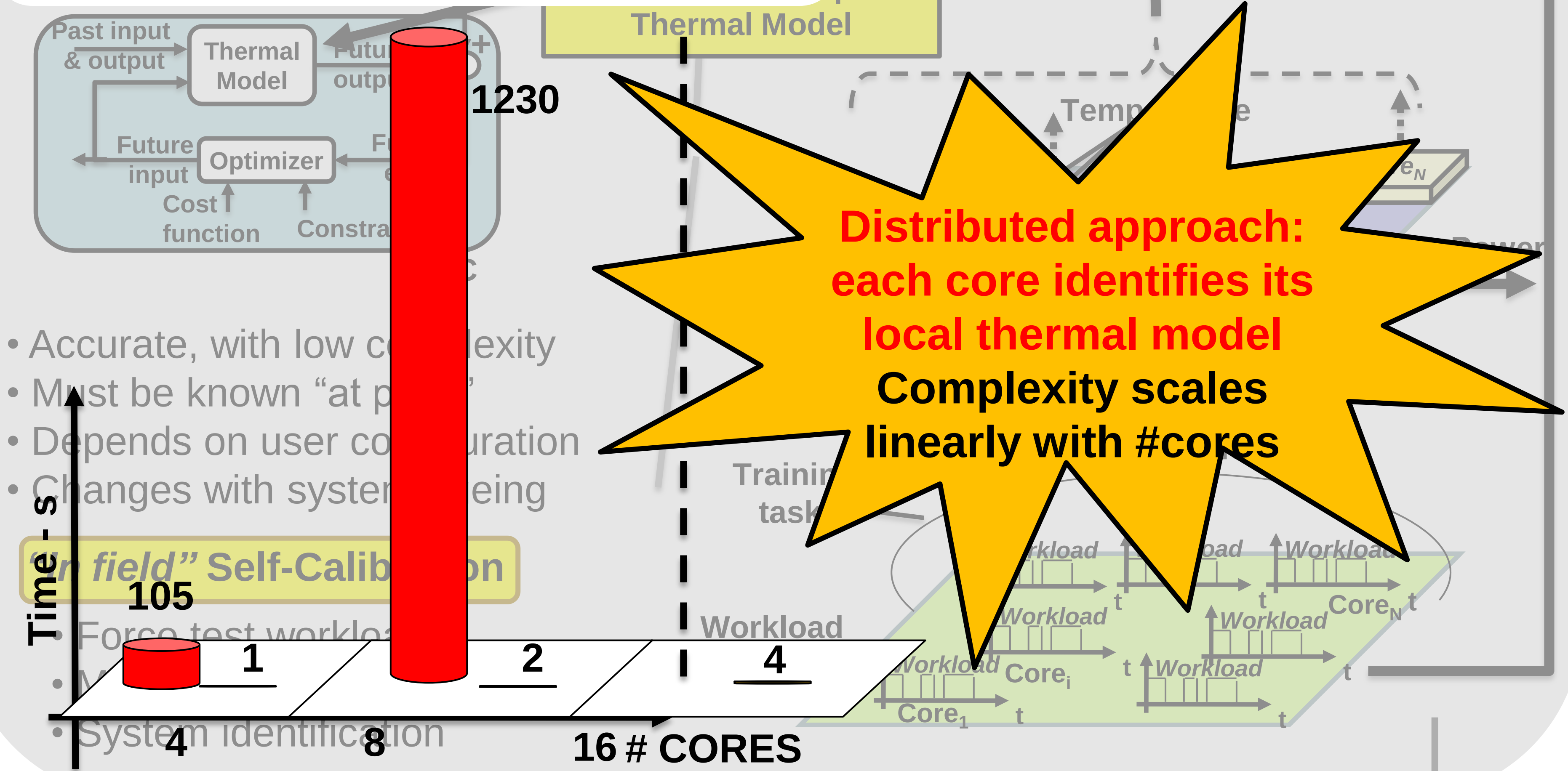
- Controller routines
  - Scheduler Routine Extension
    - Is distributed
    - Executes on the core it relies on
  - Timing: Scheduler tick (1-10ms)
- CPI estimation
  - Performance counters:
    - Clock expired
    - Instructions retired
- Energy Controller
  - Look-up-table:
    - **f_{EC} = LuT [CPI]**
- Thermal Controller
  - Core Temperature Sensors
  - Matrix Multiplication & Look-up-table:
    - **f_{TC} = LuT [M*[T_{CORE}, $T_{NEIGHBOURS}$]]**



Model Learning Scalability

Complexity issue

- State-of-the-art is centralized
- Least square based – is based on matrix inversion (cubic with #cores)



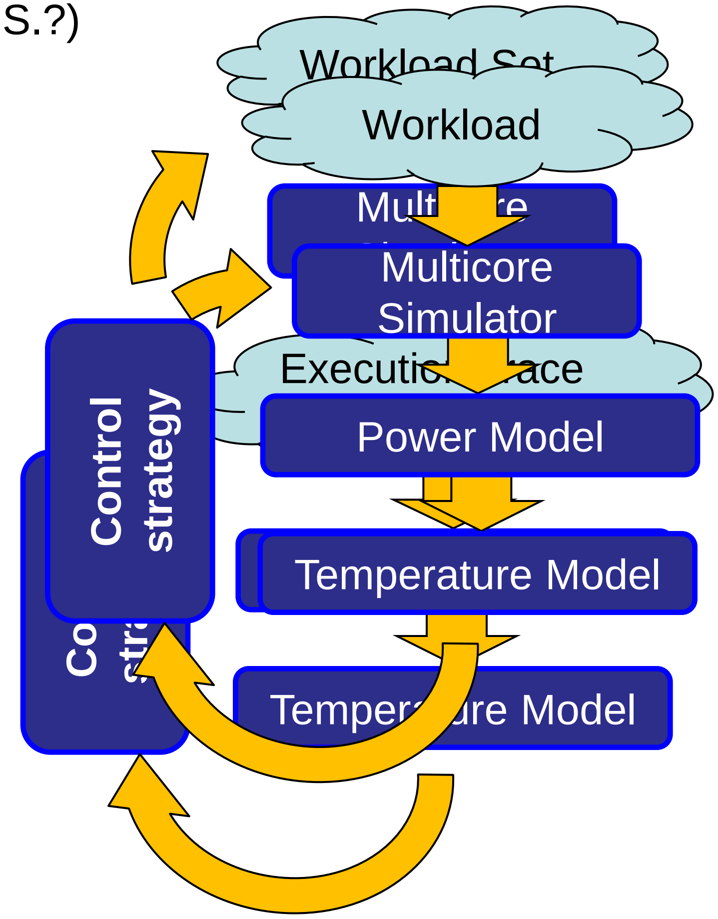
Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

Simulation Strategy

Trace driven Simulator [1]:

- Not suitable for full system simulation (How to simulate O.S.?)
- looses information on cross-dependencies
  → resulting in degraded simulation accuracy

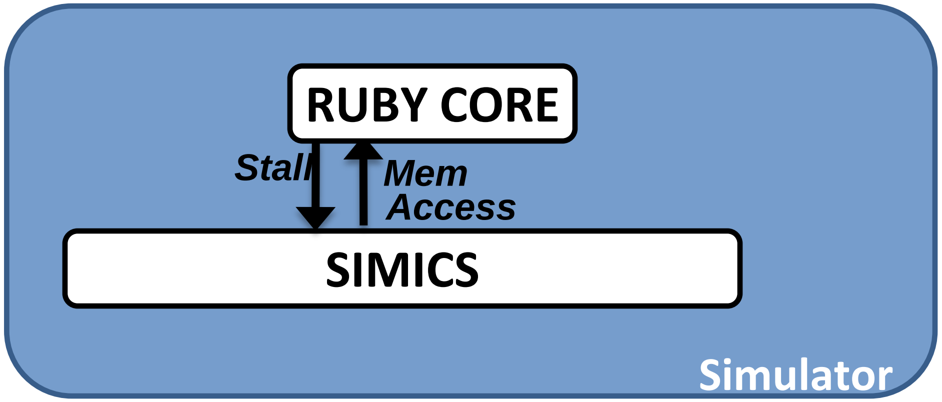


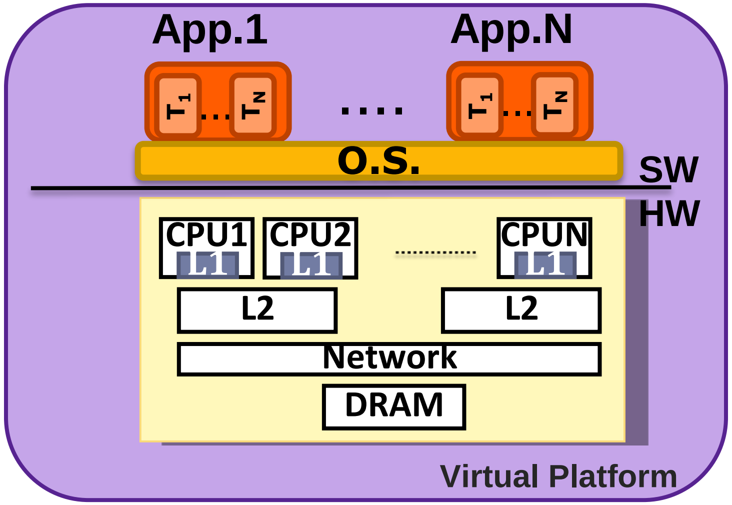
Close loop simulator:

- Cycle accurate simulators [2] :
  - High modeling accuracy
  - support well-established power and temperature co-simulation based on analytical models and system micro-architectural knowledge
  - Low simulation speed
  - Not suitable for full-system simulation
- Functional and instruction set simulators:
  - allow full system simulation
  - less internal precision
  - less detailed data → no micro-architectural model
  - introduces the challenge of having accurate power and temperature physical models

[1] P Chaparro et al. Understanding the thermal implications of multi-core architectures. *2007*
[2] Benini L. et al. MPARM: Exploring the multi-processor SoC design space with SystemC 2005

Virtual Platform





Simics by Virtutech:

- full system functional simulator
- models the entire system: peripherals, BIOS, network interfaces, cores, memories
- allows booting full OS, such as Linux SMP
- supports different target CPU (arm, sparc, x86)
- x86 model:
  - in-order
  - all instruction are retired in 1 cycle
  - does not account for memory latency

Memory timing model

- RUBY – GEMS (University of Wisconsin)[1]
  - Public cycle-accurate memory timing model
  - Different target memory architectures
  - fully integrated with Virtutech Simics
  - written in C++
  - we use it as skeleton to apply our add-ons (as C++ object)

[1] Martin Milo M. K. et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset 2005

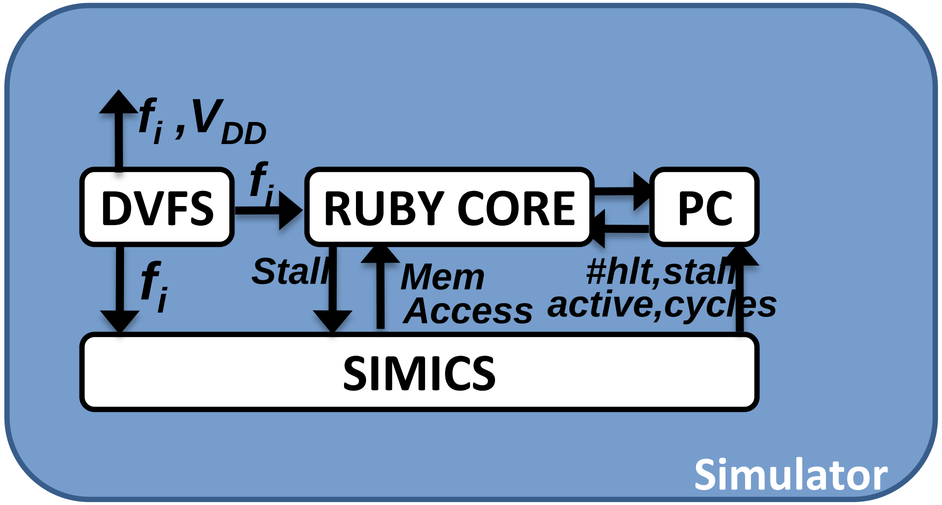
Virtual Platform

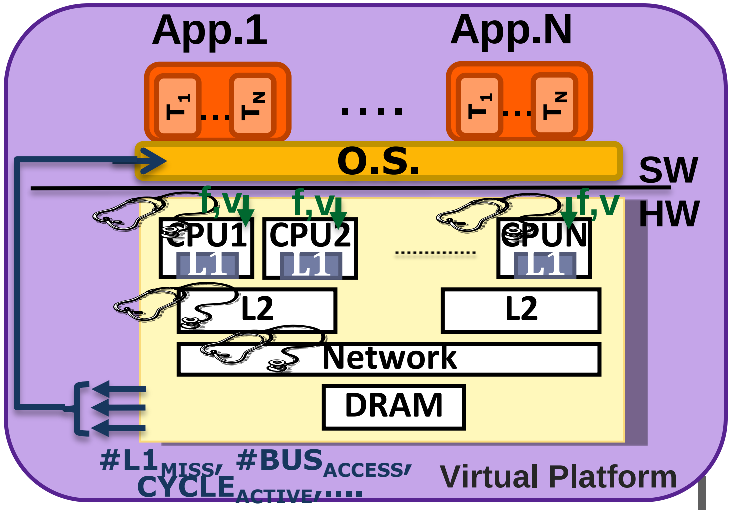
Performance knobs (DVFS) module:

- Virtutech Simics support frequency change at run-time
- RUBY does not support it:
  - does not have an internal knowledge of frequency
- We add a new DVFS module to support it
  - ensure L2 cache and DRAM to have a constant clock frequency
  - L1 latency scale with Simics processor clock frequency

Performance counters module:

- Needed by performance control policy
- We add a new Performance Counter module to support it
  - exports to O.S. and application different quantities:
    - the number of instruction retired, clock cycles and stall cycles expired,
    - halt instructions, ...

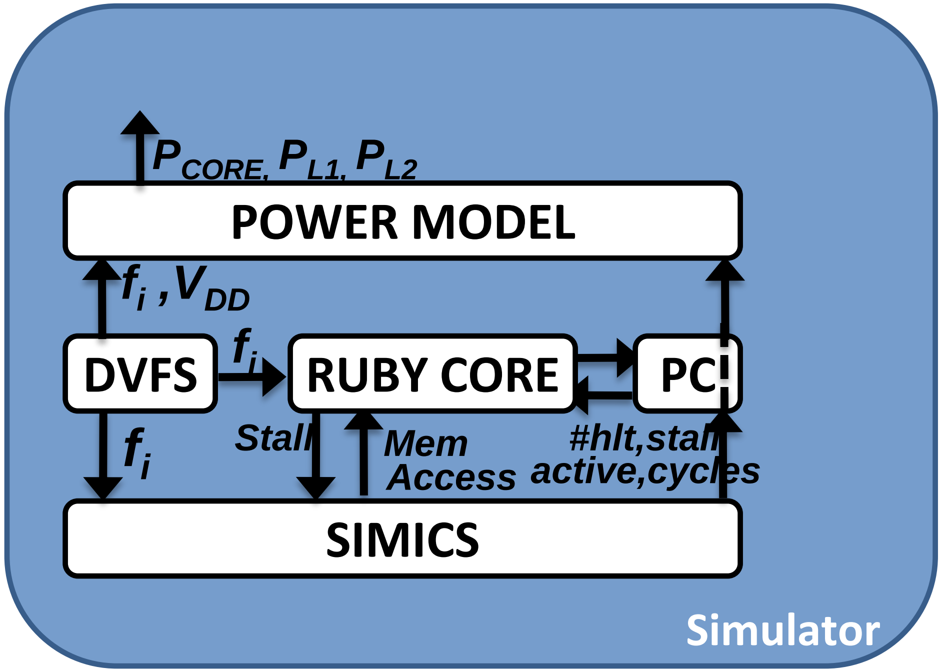


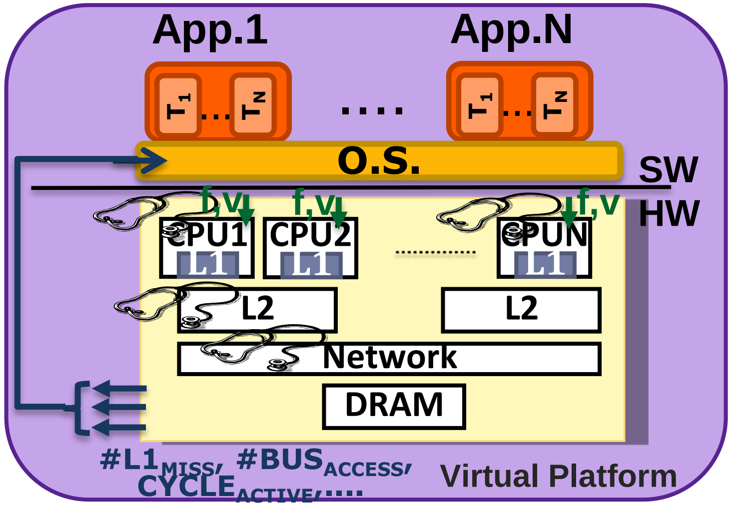


Virtual Platform

Power model module:

- At run-time estimate the power consumption of the target architecture
- Core model $P_T = [P_D(f,CPI) + P_S(T, VDD)] *(1 - idleness) + idleness *(P_{IDLE})$
- P_D experimentally calibrated analytical power model
- Cache and memory power – access cost estimated with CACTI [1]

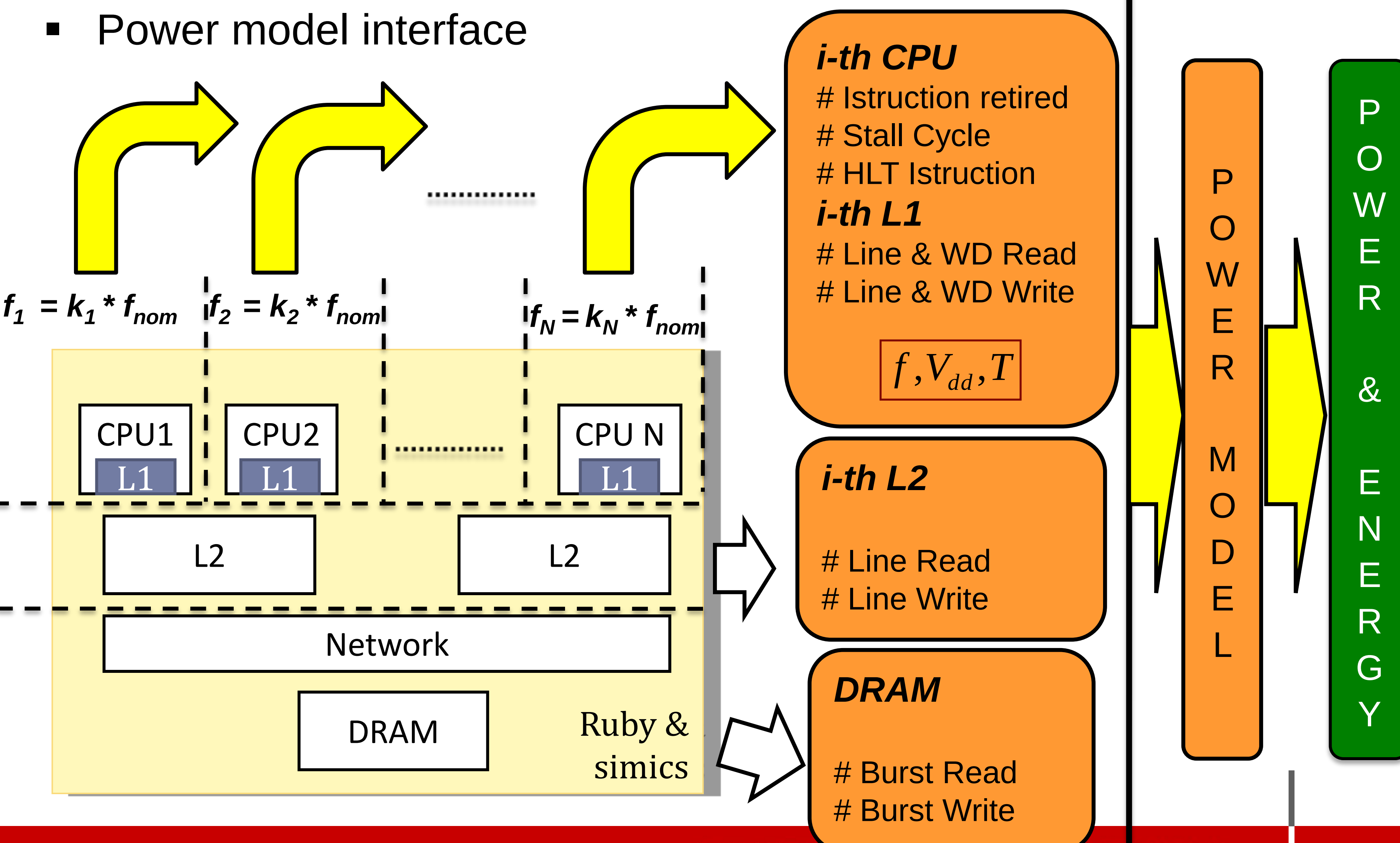




[1] Thoziyoor Shyamkumar et al. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. 2008

Power Model

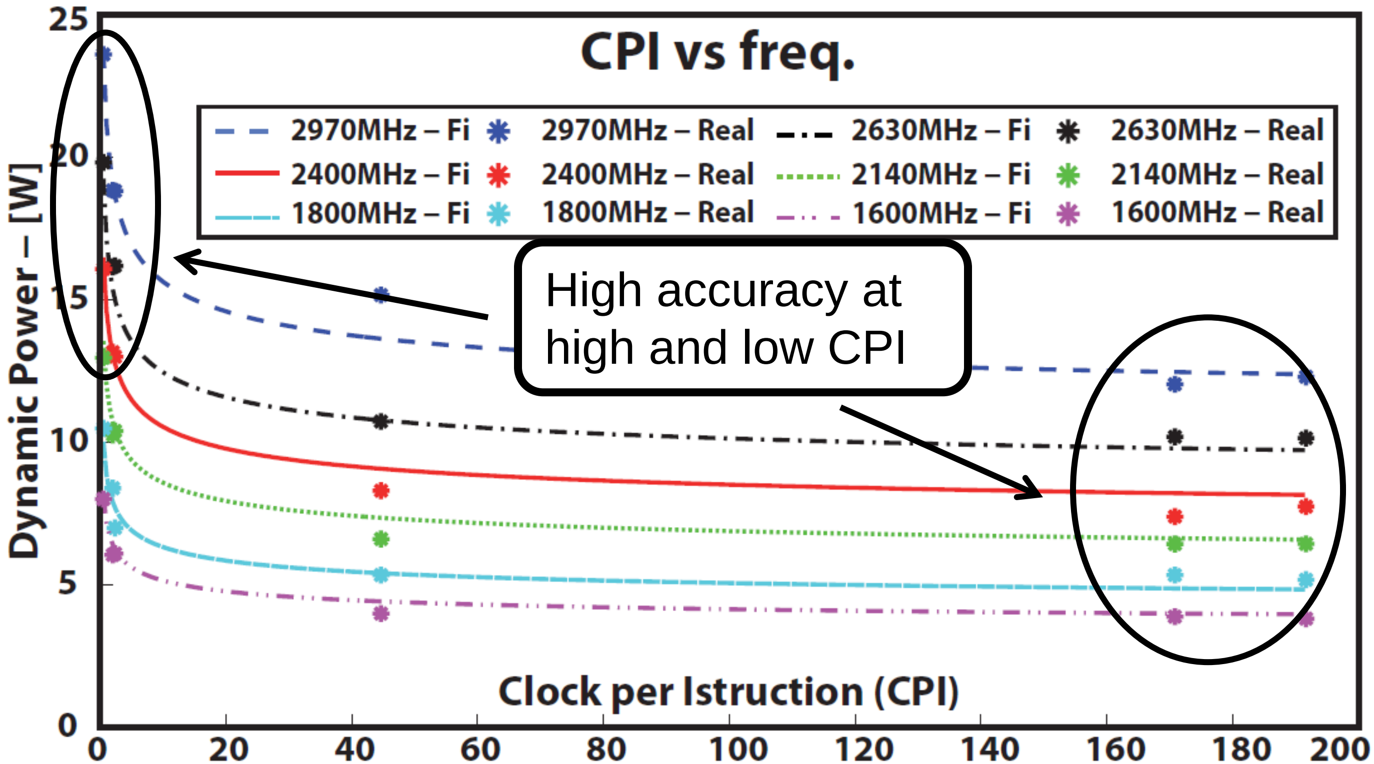
- Power model interface



Modeling Real Platform – Power

Real Power Meası

- *Intel server system*
  - 16 cores - 4 qu
  - 16GB FBDIMM
  - Intel® Core™ .
- *At the wall Power c*
  - test:
    - set of syntl
    - forcing all 1
    - for each be
    correlate it v



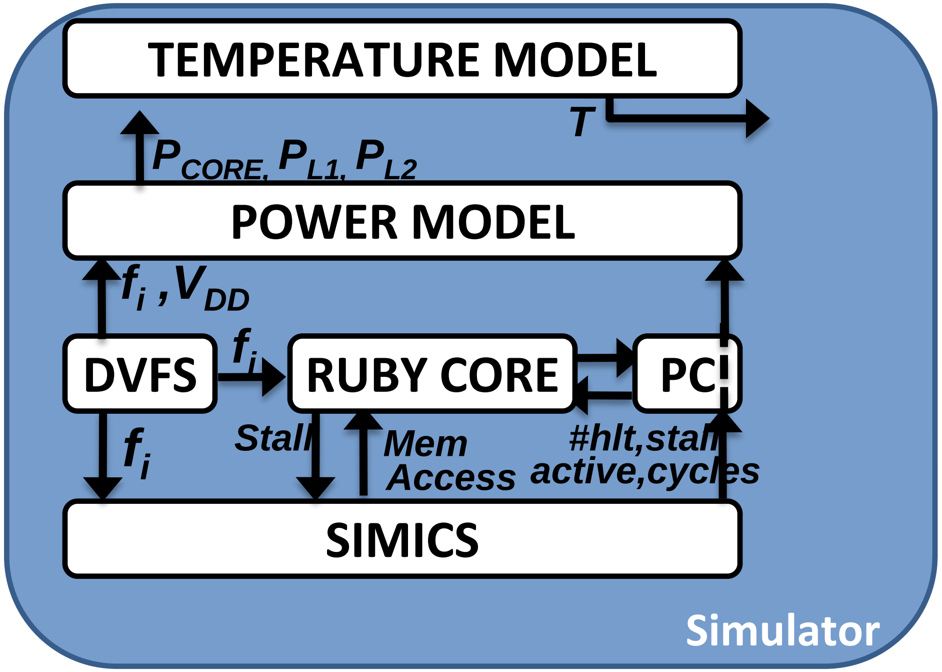
$$P_D = k_A \cdot V_{DD}^2 \cdot f_{CK} + k_B + (k_C + k_D \cdot f_{CK}) \cdot CPI^{k_E}$$

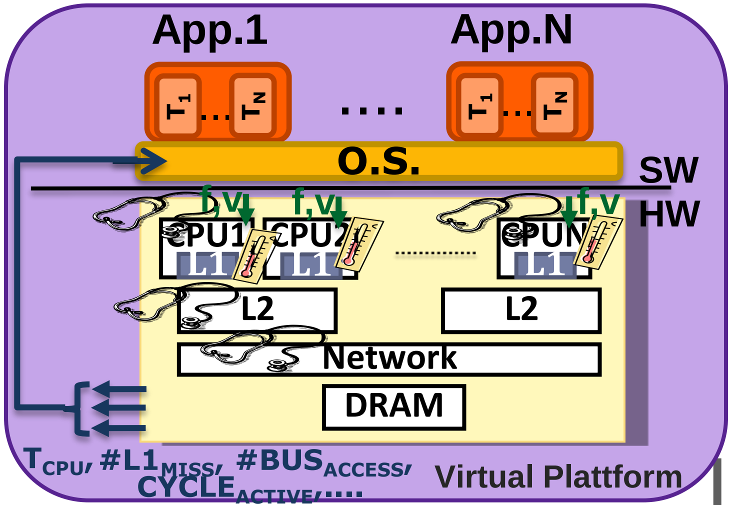
- We relate the static power with the operating point by using an analytical model

Virtual Platform

Temperature model module:

- we integrate our virtual platform with a thermal simulator [1]
- Input: power dissipated by the main functional units composing the target platform
- Output: Provides the temperature distribution along the simulated multicore die area as output

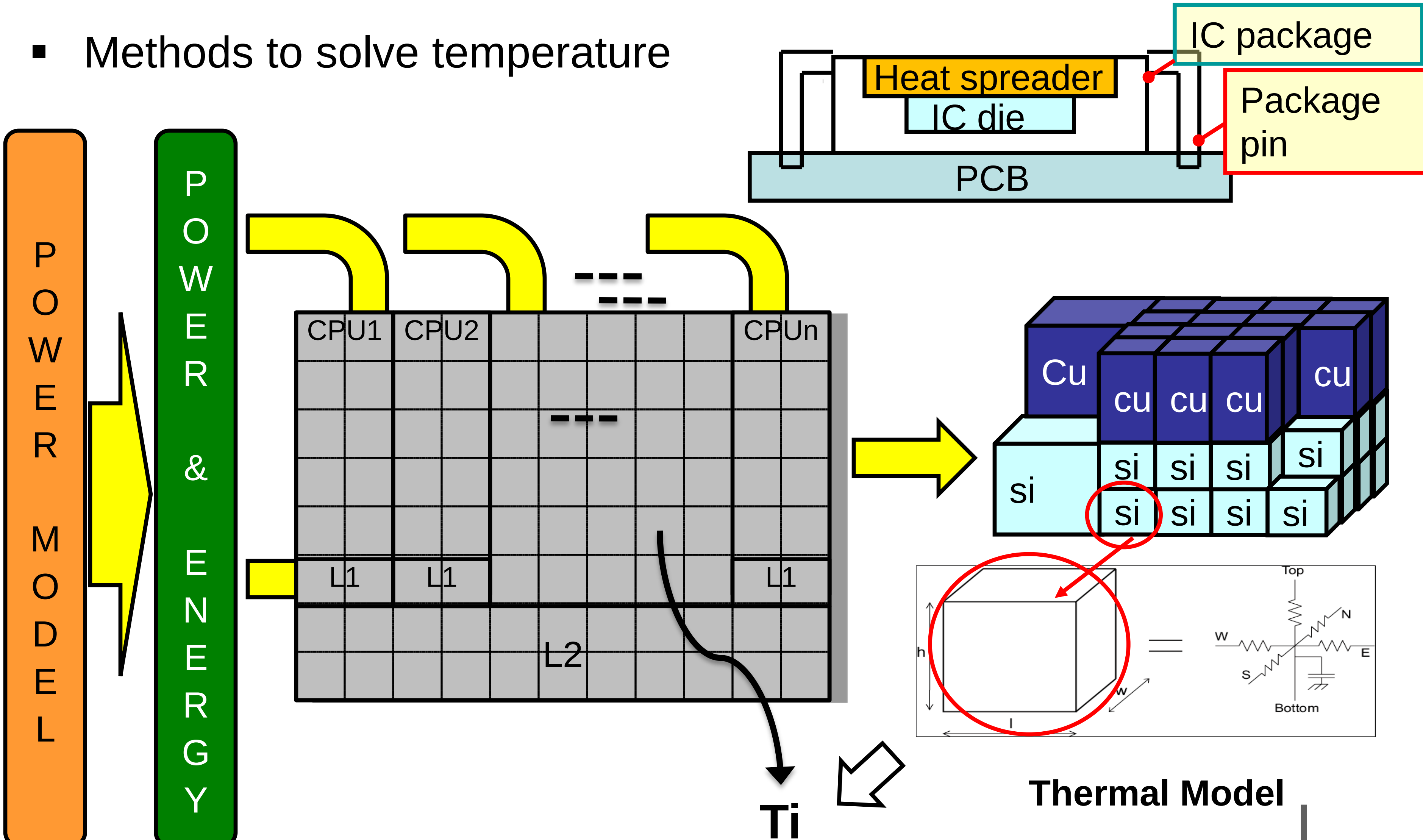




[1] Paci G. et al. Exploring "temperature-aware" design in low-power MPSoCs

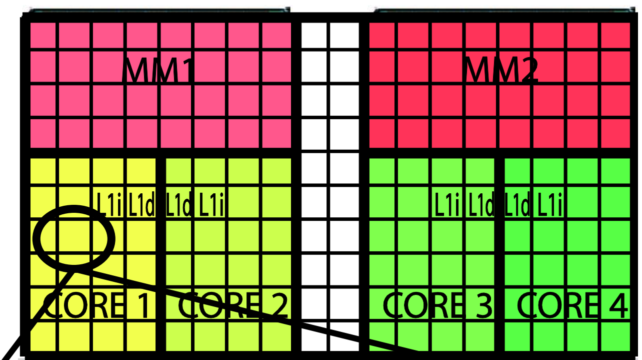
Thermal Model

- Methods to solve temperature

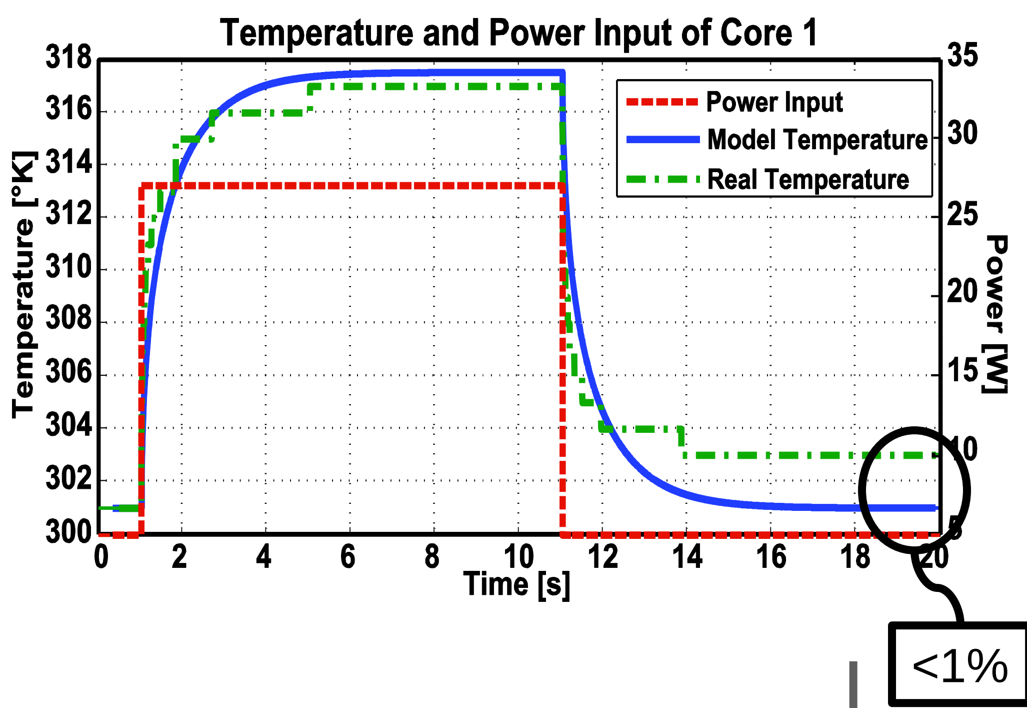


Modeling Real Platform– Thermal

- Thermal Model Calibration :
  - Derived from Intel® Core™ 2 Duo layout
  - We calibrate the model parameter to simulate real HW transient
  - High accuracy (error < 1%) and same transient behavior



silicon thermal conductivity	$150 \cdot (\frac{300}{T})^{4/3}$ W/mK
silicon specific heat	$1.628e^{-12}$ J/um
silicon thickness	350um
copper thermal conductivity	400W/mK
copper specific heat	$3.55e^{-12}$ J/um 3K
copper thickness	2057um
elementary cell length	1312um
package-to-air conductivity	0.4K/W



Virtual Platform Performance

- Target:
  - 4 core Pentium® 4
  - 2GB RAM
  - 32 KB private L1 cache
  - 4 MB shared L2 cache
  - Linux OS
- Host:
  - Intel® Core™ 2 Duo
  - 2.4 Ghz
  - 2GB RAM

1 Billion instruction

Simics + Ruby:
Tsim = 1040 s

Simics + Ruby + DVFS:
Tsim = 1045 s

Simics + Ruby + DVFS + Power:
Tsim = 1110 s
Compute every 13us

+ 7%

Simics + Ruby + DVFS + Power + Thermal interface:
Tsim = 1160 s

Simics + Ruby + DVFS + Power + Thermal Model:
Tsim = 1240 s
68 cells
T = 100ns

+ 19.2%

Mathworks Matlab/Simulink

- **Numerical computing environment** developed to design, implement and test numerical algorithms
- Mathworks Simulink – for **simulation of dynamic systems**: simplifies and speedups the development cycle of control systems
- **Can be called** as **a computational engine by** writing C and Fortran **programs** that use Mathworks Matlab's engine library
- Controller design - two steps:
  - developing the control algorithm that optimizes the system performance
  - implementing it in the system

We allow a Mathworks Matlab/Simulink description of the controller to directly drive at run-time the performance knobs of the emulated system

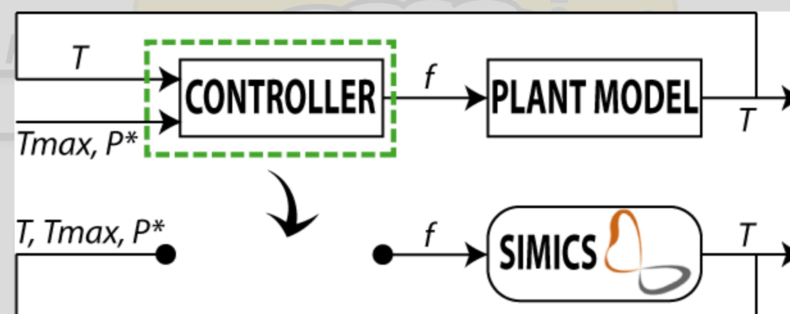
Virtual Platform

Mathworks Matlab interface:

- New module named Controller in RUBY
- Initialization: starts the Mathworks Matlab engine concurrent process,
- Every N cycle - wake-up:
  - send the current performance monitor output to the Mathworks Simulink model
  - execute one step of the controller Mathworks Simulink model
  - propagate the Mathworks Simulink controller decision to the DVFS module

CONTROL-STRATEGIES DEVELOPMENT CYCLE

1. Controller design in Mathworks Matlab/Simulink framework
   - system represented by a simplified model
   - obtained by physical considerations and identification techniques
2. Set of simulation tests and design adjustments done in Simulink
3. Tuned controller evaluation with an accurate model of the plant done in the virtual platform



4. Performance analysis, by simulating the overall system

Outline

- Introduction
- Energy Controller
- Thermal Controller architecture
- Learning (self-calibration)
- Scalability
- Simulation Infrastructure
- Results
- Conclusion

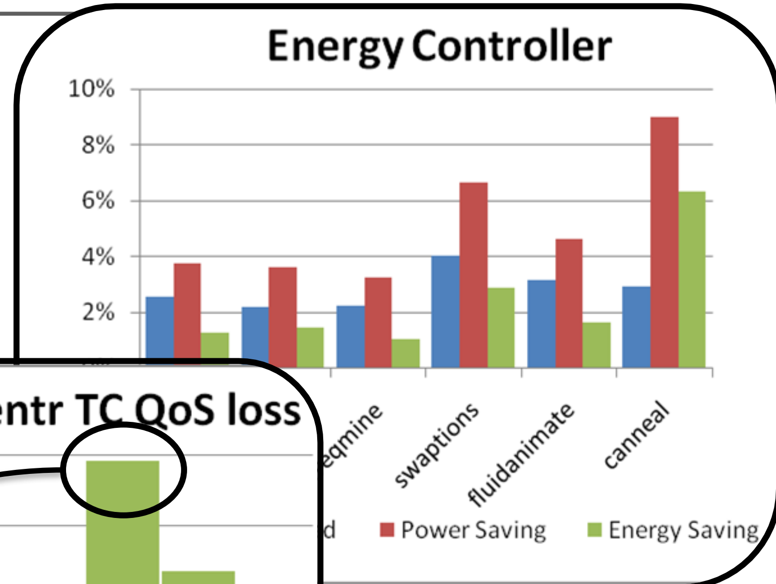
Results

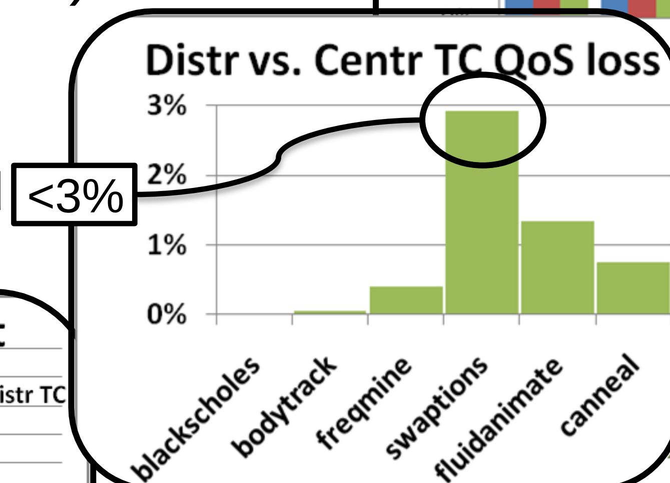
Energy Controller (EC)

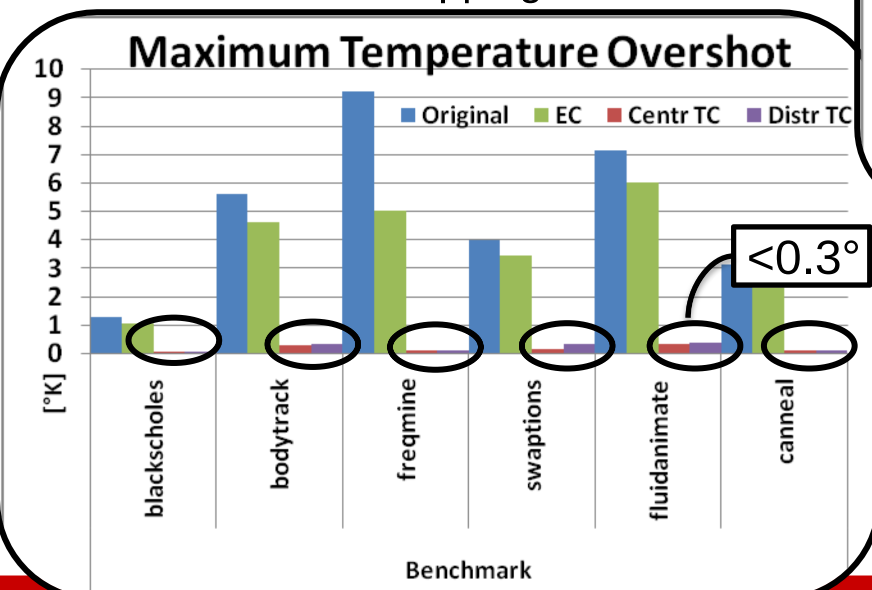
- Performance Loss < 5%
- Energy minimization

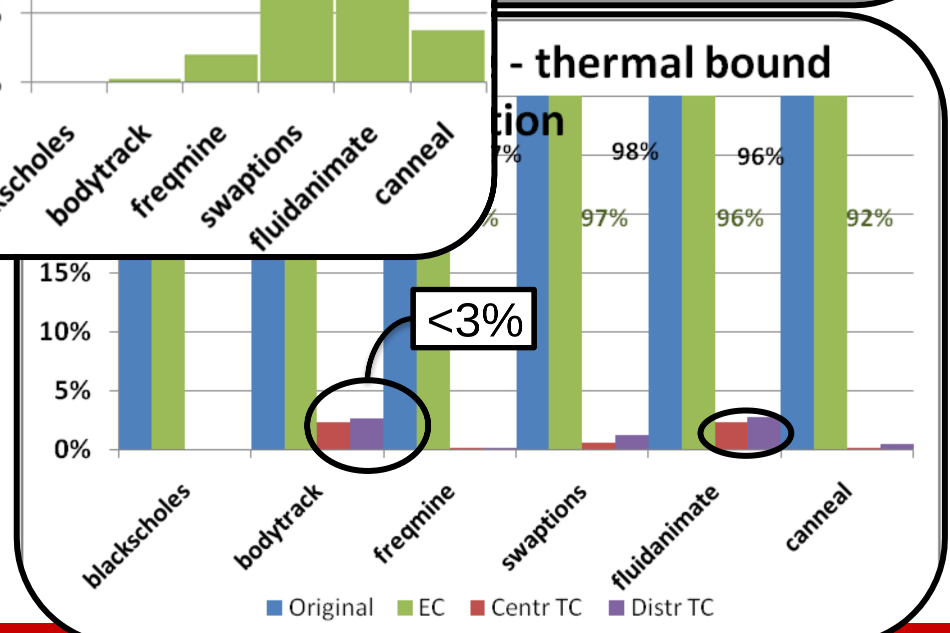
Temperature Controller (TC)

- Complexity reduction
  - 2 explicit region for controller
- Performs as the centralized
  - Thermal capping





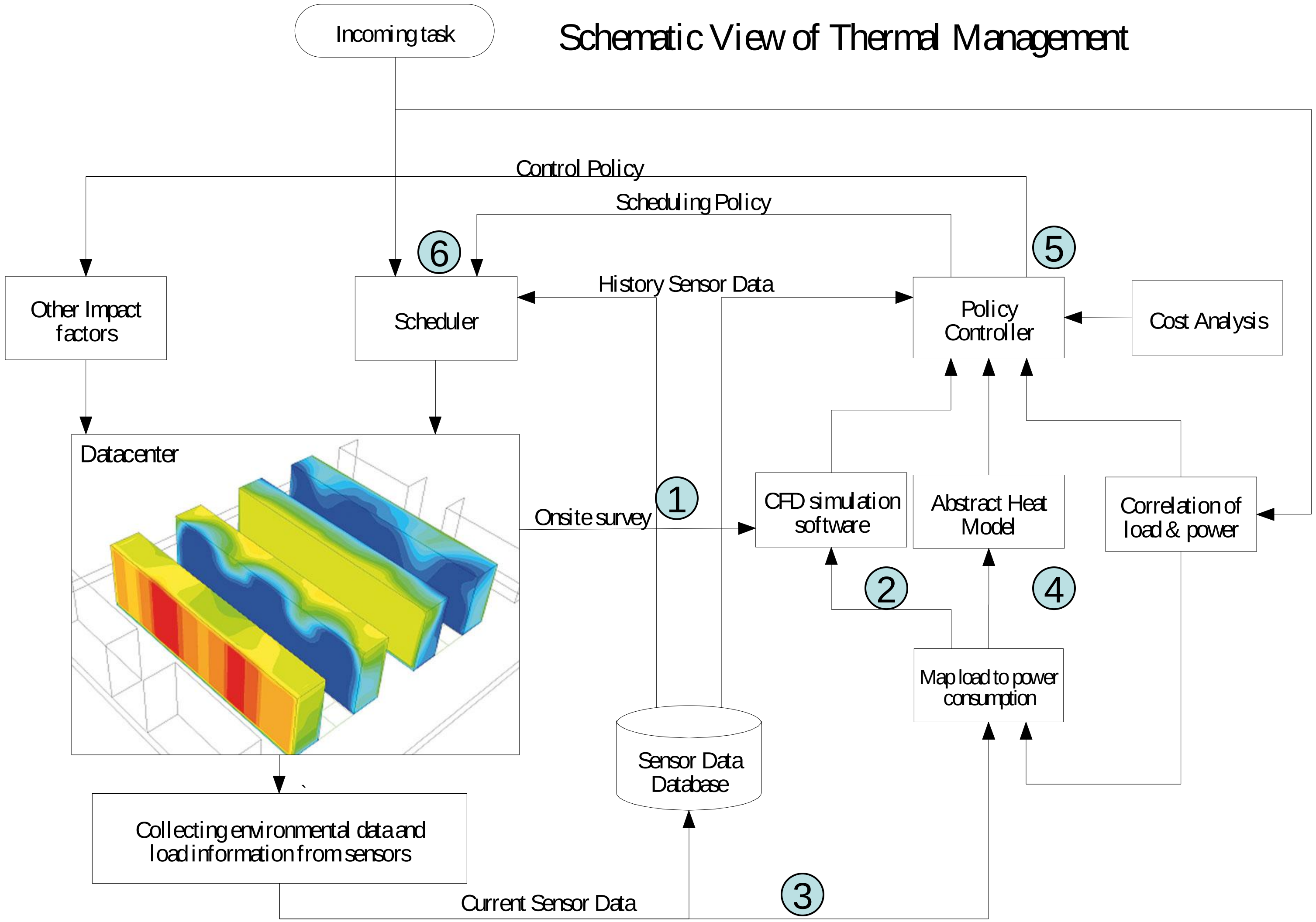




Next Steps

- Now working on the embedded implementation
  - Server multicore platform and Intel ® SCC
- Explore thermal aware scheduler solution
  - co-operate with presented solution
- Develop distributed+multi-scale solution for data-centers

Thermal-aware task scheduling



Thank you!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA