Thermac: Steps towards the thermal-aware allocation

2021-01-27

CTU in Prague

Thermal-aware Resource Management for Modern Computing Platforms in the Next Generation of Aircraft

Problem Statement

Thermal-aware allocation of SC tasks

Inputs

- set of **resources** $\mathcal{R} = \{R_1, \ldots, R_m\}$
- resource **capacity** $c_k \in \mathbb{N},\ \forall R_k \in \mathcal{R}$
- set of (SC) **tasks** $\mathcal{T} = \{T_1, \ldots, T_n\}$
- tasks' characteristics,
  e.g., **processing time** $p_{i,k} \in \mathbb{N}^+ \cup \{\infty\},\ \forall T_i \in \mathcal{T}, R_k \in \mathcal{R}$
- major frame **length** h

Each resource represents one computational cluster (e.g., A53 or A72 cluster). The capacity represents the number of the computational units (cores) of the resource. For our purposes, the capacities are 4 and 2, respectively.

Assumptions

- all (SC) tasks are ready at the beginning of the period
- all tasks need to be scheduled within the major frame
- scheduling of the SC tasks is non-preemptive
- each SC task needs to be finished inside of its assigned window
- the tasks are single threaded (single-core)
- the major frame (schedule) is executed repeatedly
- length of the major frame h is much smaller than the time constants of the system
- the frequency of the cores is fixed

Goal

Find the **isolation windows** $\mathcal{W} = \{W_1, \ldots, W_\ell\}$ and their lengths $l_j \in \mathbb{N}_0^+, \ \forall W_j \in \mathcal{W}$. Also, find the **window assignment** function $a_w : \mathcal{T} \to \mathcal{W}$ and **resource assignment** function $a_r : \mathcal{T} \to \mathcal{R}$.

The solution $(\ell, \boldsymbol{l}, a_w, a_r)$ is feasible if

- the length of all windows is at most h

$$\sum_{W_j \in \mathcal{W}} l_j \leq h \tag{1}$$

- at most 60 % of each window (each core) is filled by SC tasks

$$l_j \geq \frac{\max\{p_{i,k} \mid T_i \in \mathcal{T} \wedge a_w(T_i) = W_j \ \wedge \ a_r(T_i) = R_k\}}{0.6} \ \forall W_j \in \mathcal{W} \tag{2}$$

- the number of assigned tasks to each cluster is lower or equal to its capacity

$$\sum_{T_i \in \mathcal{T} : a_w(T_i) = W_j} \mathbb{1}_{[a_r(T_i) = R_k]} \leq c_k, \ \forall W_j \in \mathcal{W}, R_k \in \mathcal{R} \tag{3}$$

Example

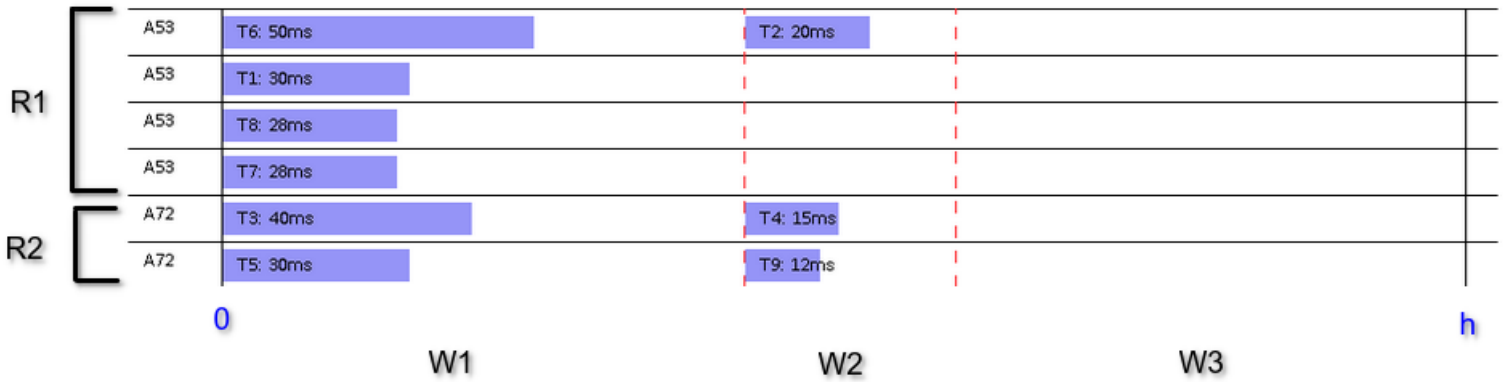


Figure: Example of a schedule with 3 windows and 9 tasks.

Best-effort tasks

What to measure/optimize

the steady-temperature or the average power

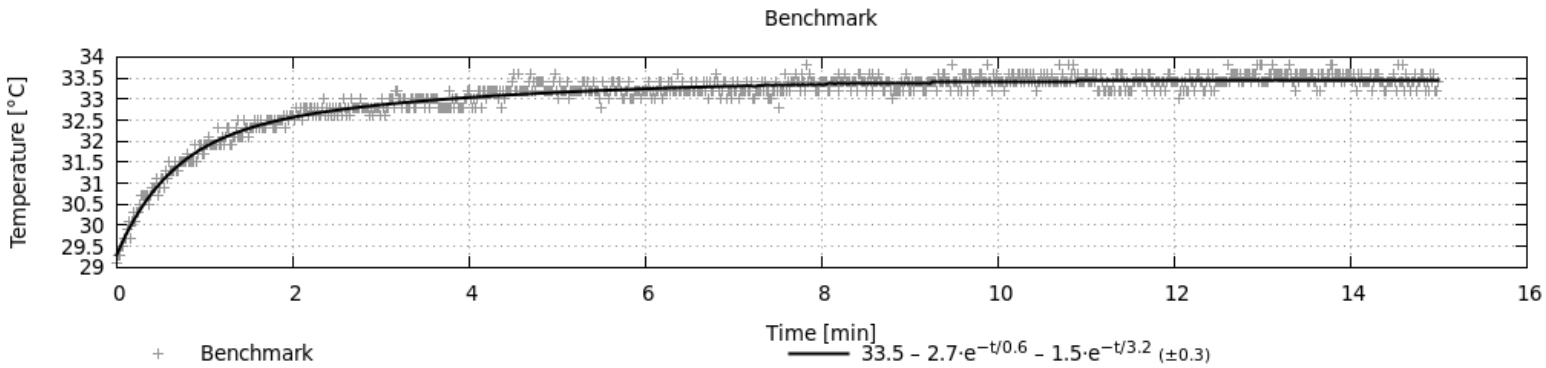
Objective

Find such solution $(\ell, \boldsymbol{I}, a_w, a_r)$ that will minimize the **steady-state temperature** of the platform while being executed repeatedly.

So far, we have approximated the behavior of the platform by 'sum-of-exponentials' model.

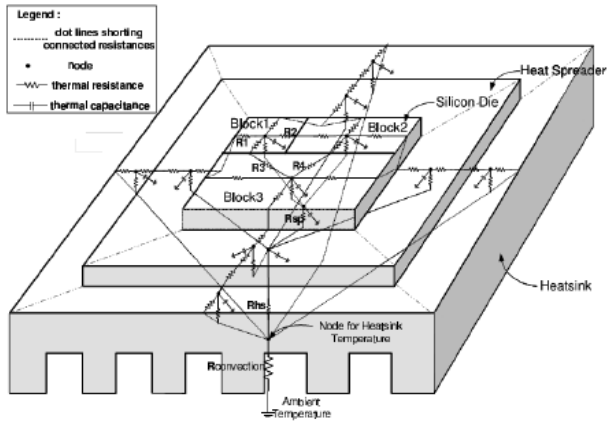
$$T_n(t) = T_\infty + \sum_{i=1}^{n} \kappa_i e^{-\frac{t}{\tau_i}} \tag{4}$$

The model is fitted to the measured data and the steady-state temperature T_∞ is then extracted.

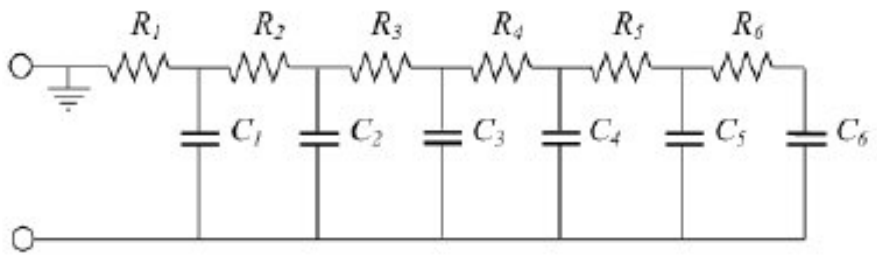


Modeling in a broader perspective

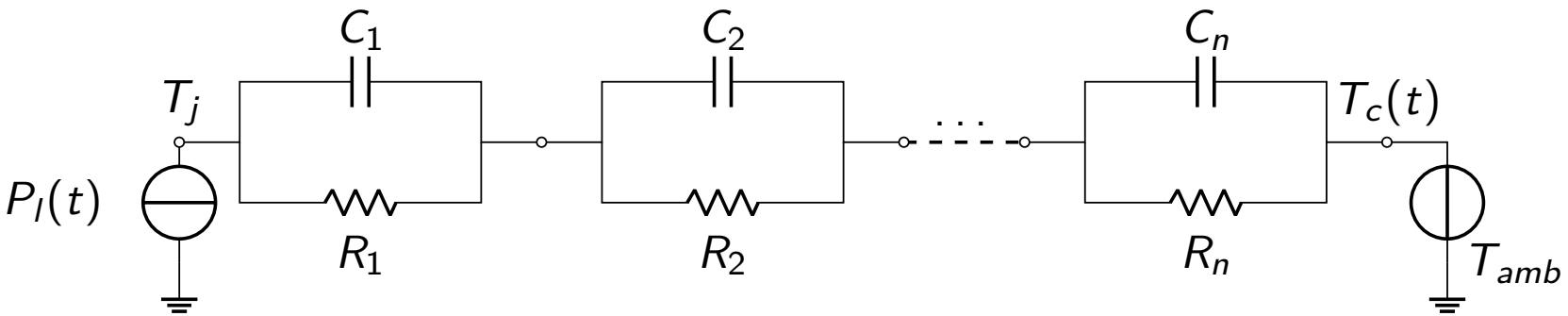
- **RC network** models suited for multi-input multi-output thermal system. The disadvantage is the complexity of the modeling/simulations. Also, identification of the parameters is not trivial.



- **RC ladder** approach models only a single conduction path. For MIMO systems (chip with multiple hotspots) results of several models need to be superimposed. This approach is therefore suitable for systems with limited number of points of interest.



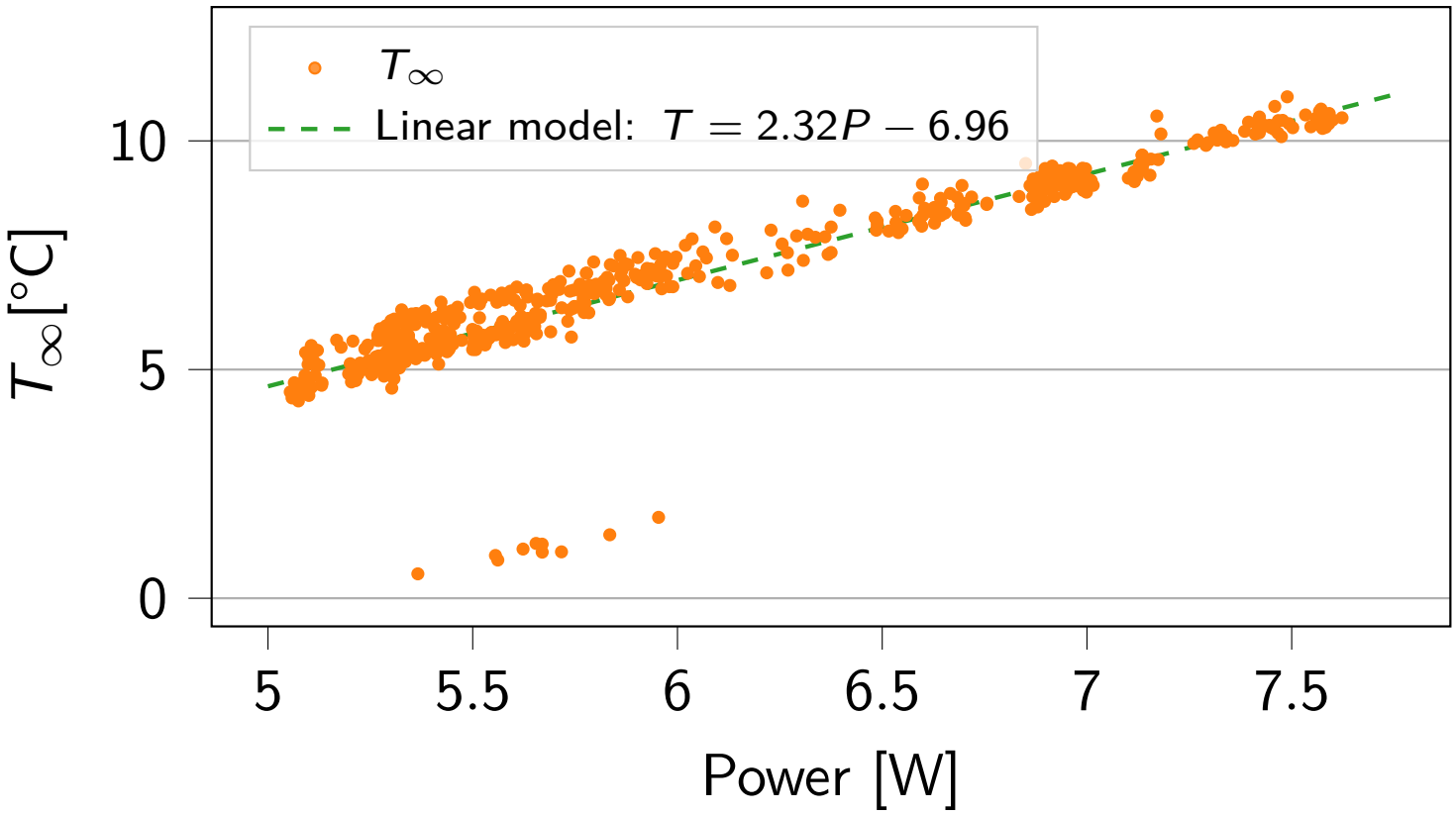
Sum-of-exponentials model in context of RC ladders



- Modeling the thermal behavior of the system by **Foster RC ladder**.
- **Thermal impedance** (i.e., the difference in temperature between two isothermal surfaces divided by the rate of heat flowing across the hotter isothermal boundary; here $Z_{jc} = \frac{(T_j - T_c)}{P}$) – for Foster's ladder, we have $Z(t) = \sum_{i=1}^{n} R_i \left(1 - e^{-\frac{t}{R_i C_i}}\right)$
- For the steady-state, the thermal impedance will correspond to the thermal resistance being $Z(\infty) = \sum_{i=1}^{n} R_i$. In sum-of-exponentials model, we used term T_∞, which corresponds to $P \sum_{i=1}^{n} R_i$ in this settings (assuming a constant power input P). Hence we can assume $T_\infty \simeq P$.

Power and temperature relation

- Multiple experiments (various CPU/memory benchmarks, different combinations of active cores) were performed at frequency 600 MHz (common for both computing clusters).



Pros and Cons

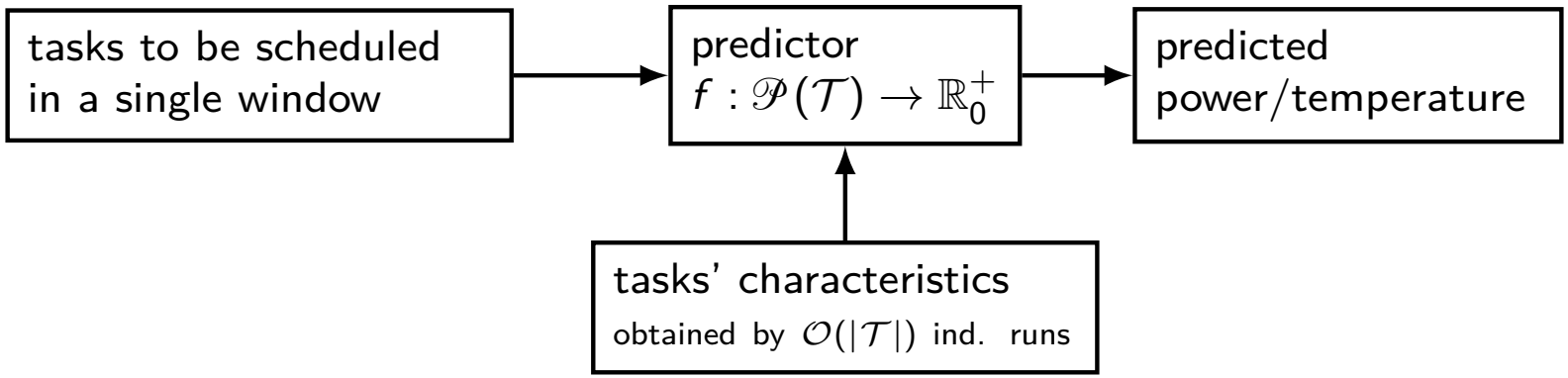
Temperature	Power
measurements take long time	measurements can be made faster
measurements depend on the ambient temperature	independent on small changes of the ambient temperature
easier to measure	harder to measure precisely (sampling rate, noise, . . .)
captures the final temperature of the whole system	simplifies the system's behavior, may not be accurate when power is not constant
	the final evaluation needs to be done while measuring the temperature anyway

Tasks' characteristics

The idea

The overall goal is to schedule the tasks within windows such that the steady-state temperature of the schedule's repeated execution is minimized.

We want to **avoid** the testing of all possible tasks' combinations on all clusters/cores. Instead, the idea is to benchmark the tasks **individually**, measure the **tasks' characteristics**, and based on that **predict** the performance of the tasks' combinations.

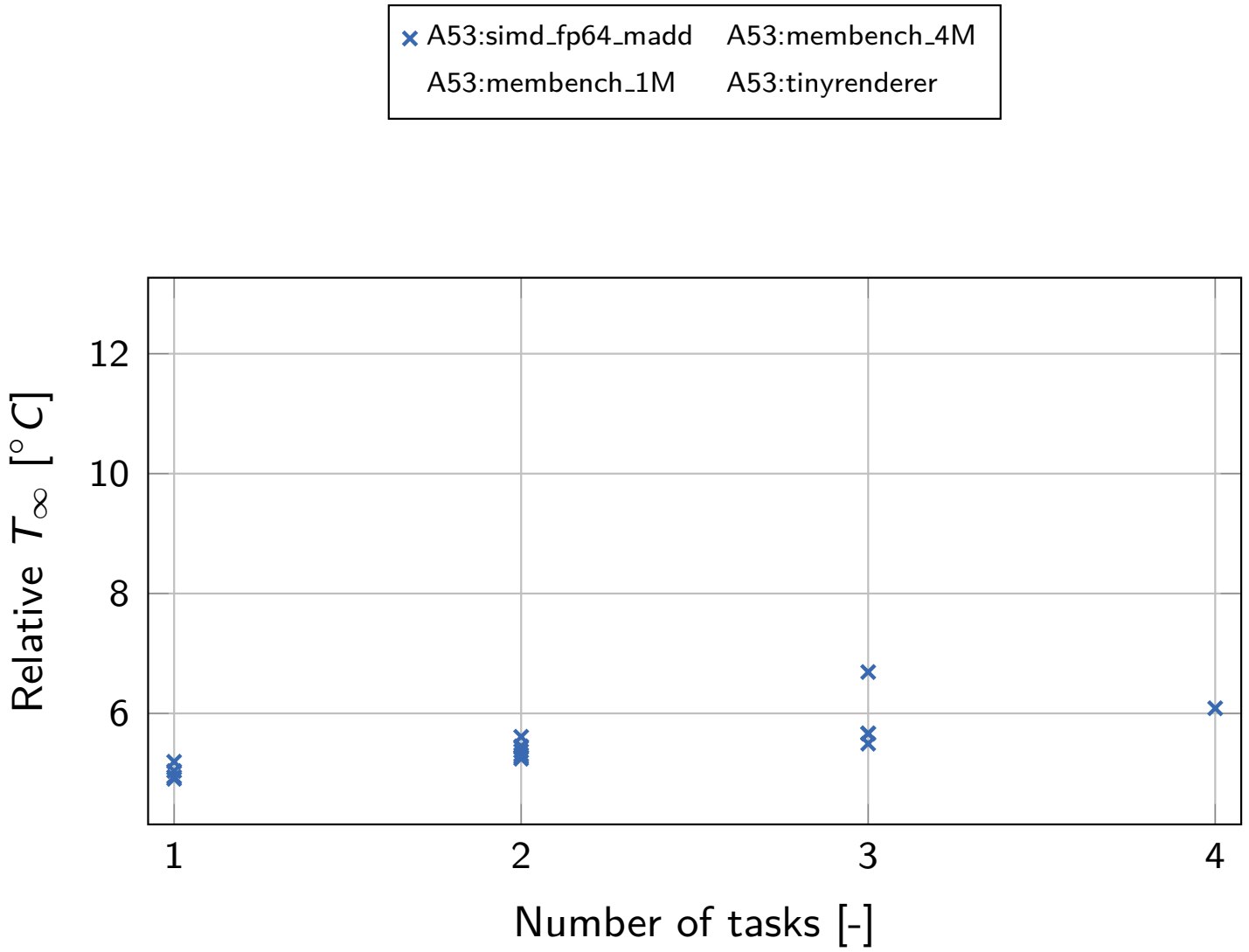


Characteristics

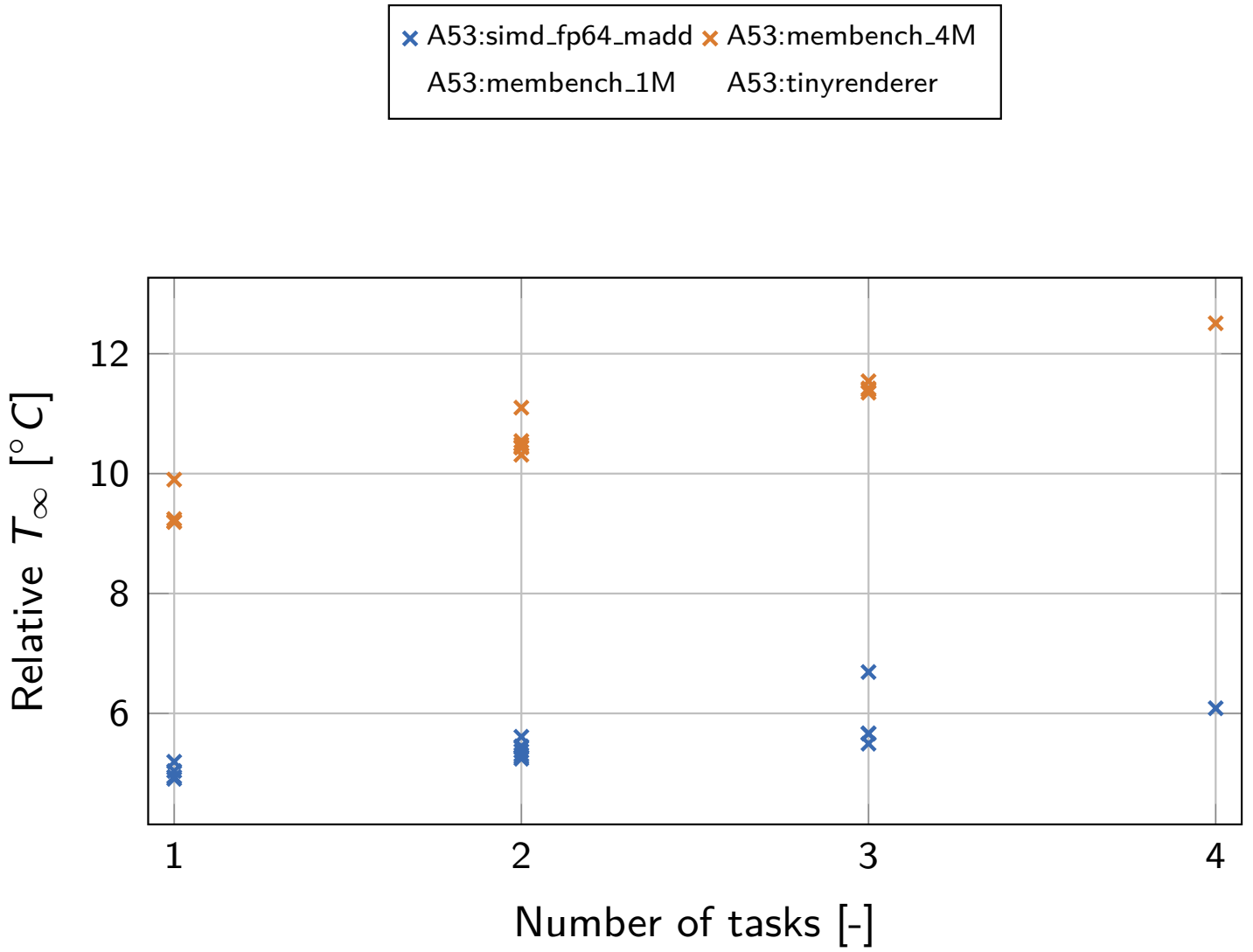
- Measure the **static** and **active** power consumption (or the corresponding temperature) of each benchmark
  - Static power – dissipated by leakage currents even when the device is inactive
  - Active power – dependent on core's switching activity

Each benchmark is executed on 1/all cores of each cluster. The measured power/temperature is fitted by a linear function $y = ax + b$. Coefficient a (slope) then represents the active power, while b (intercept) the static power.

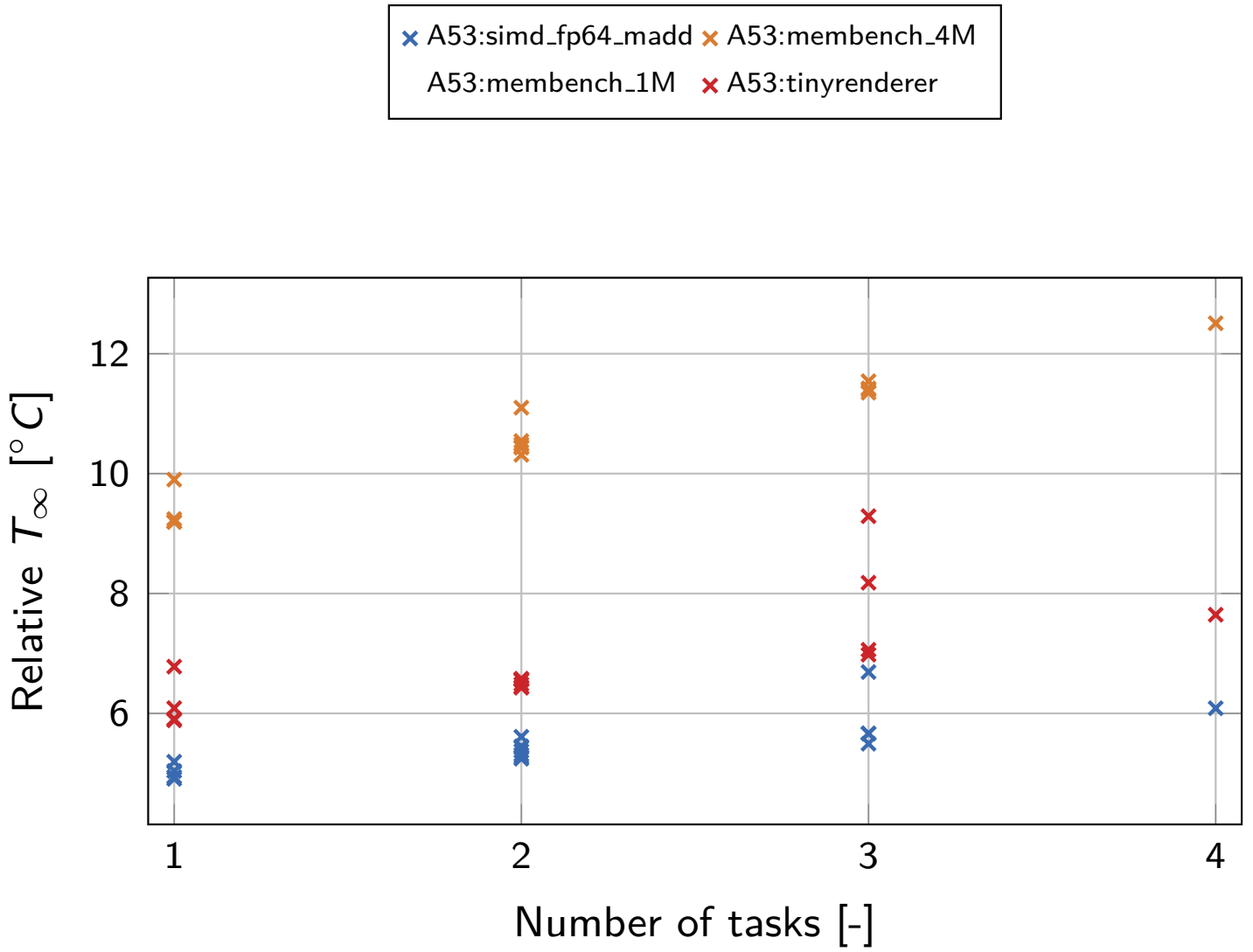
Example measurements



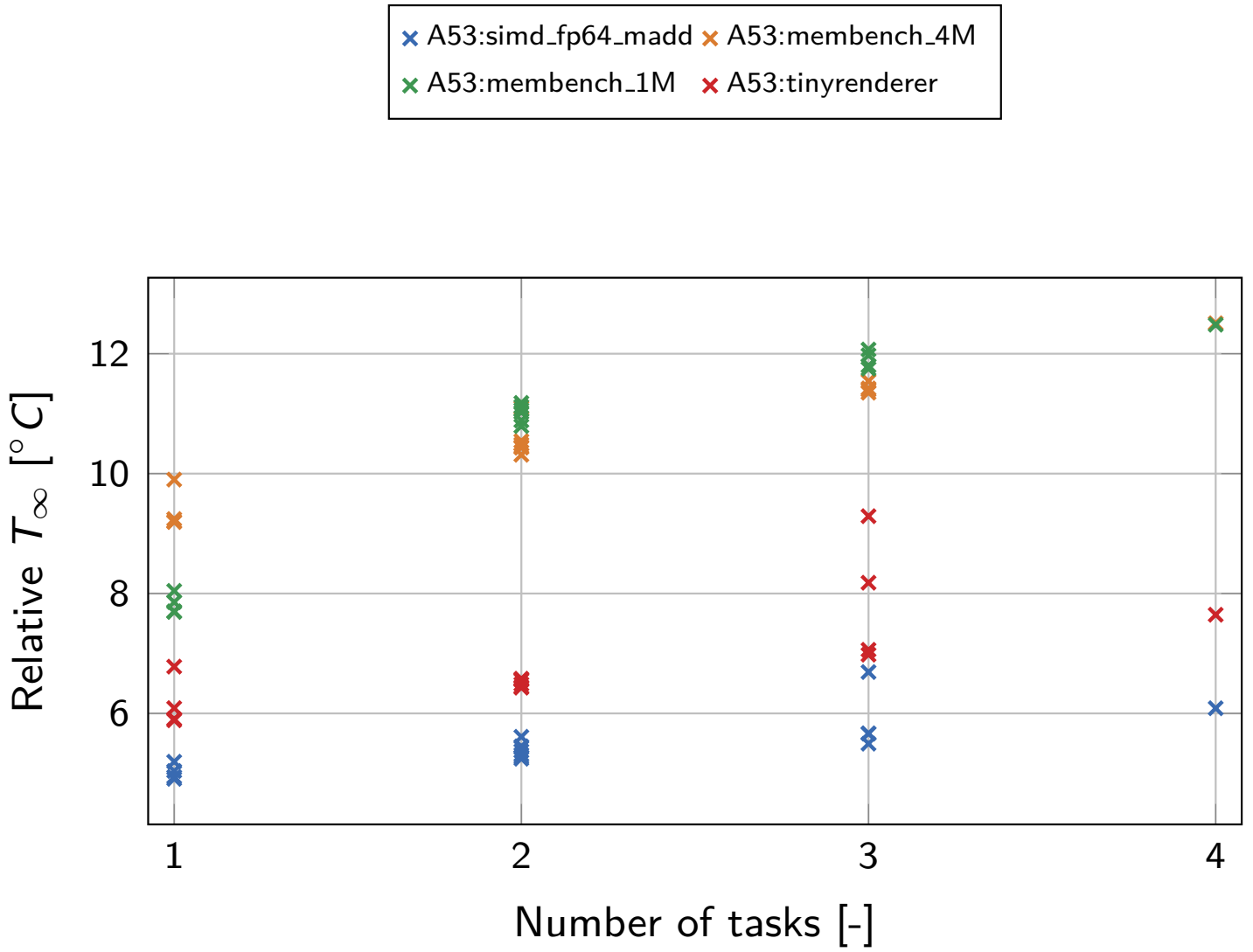
Example measurements



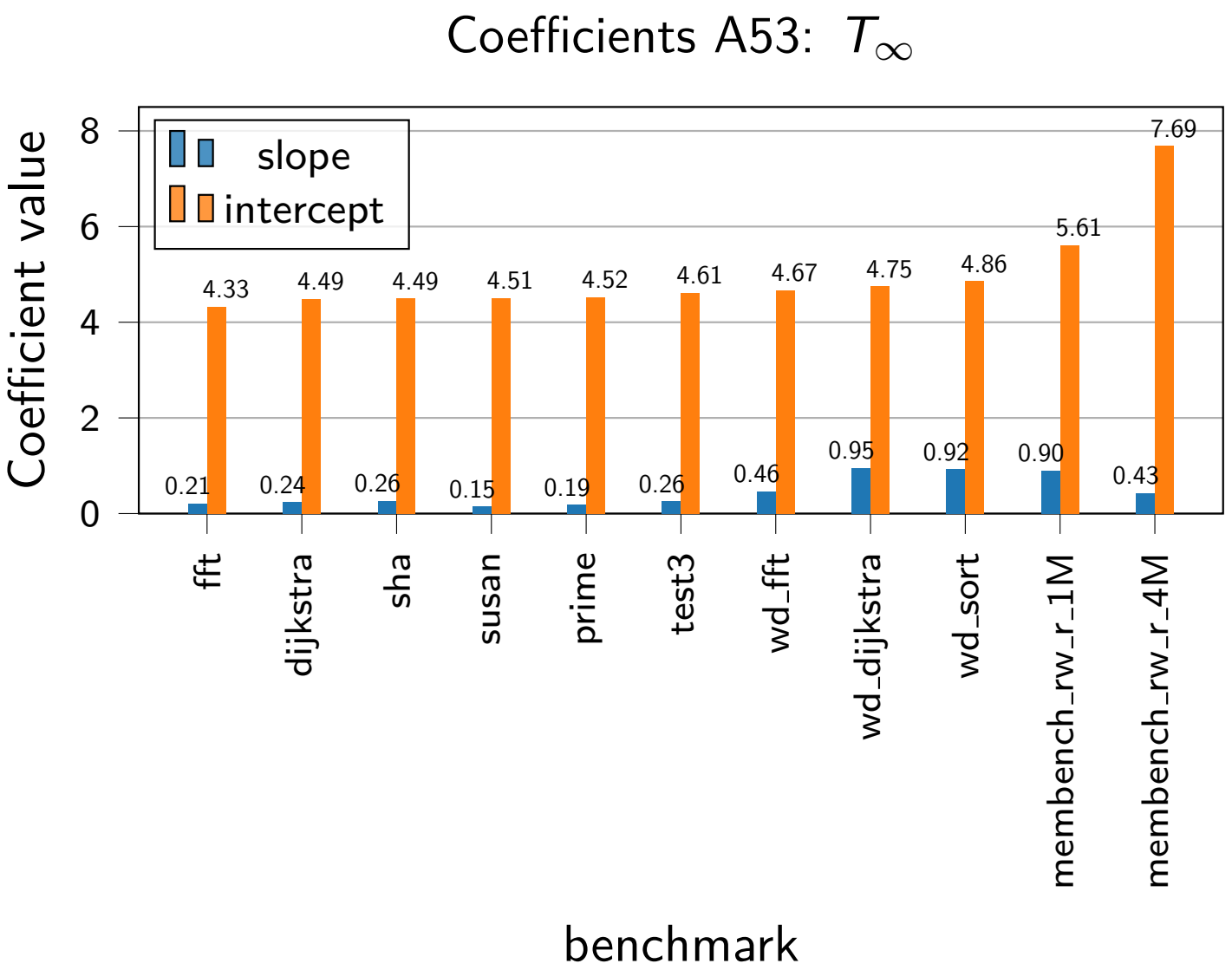
Example measurements



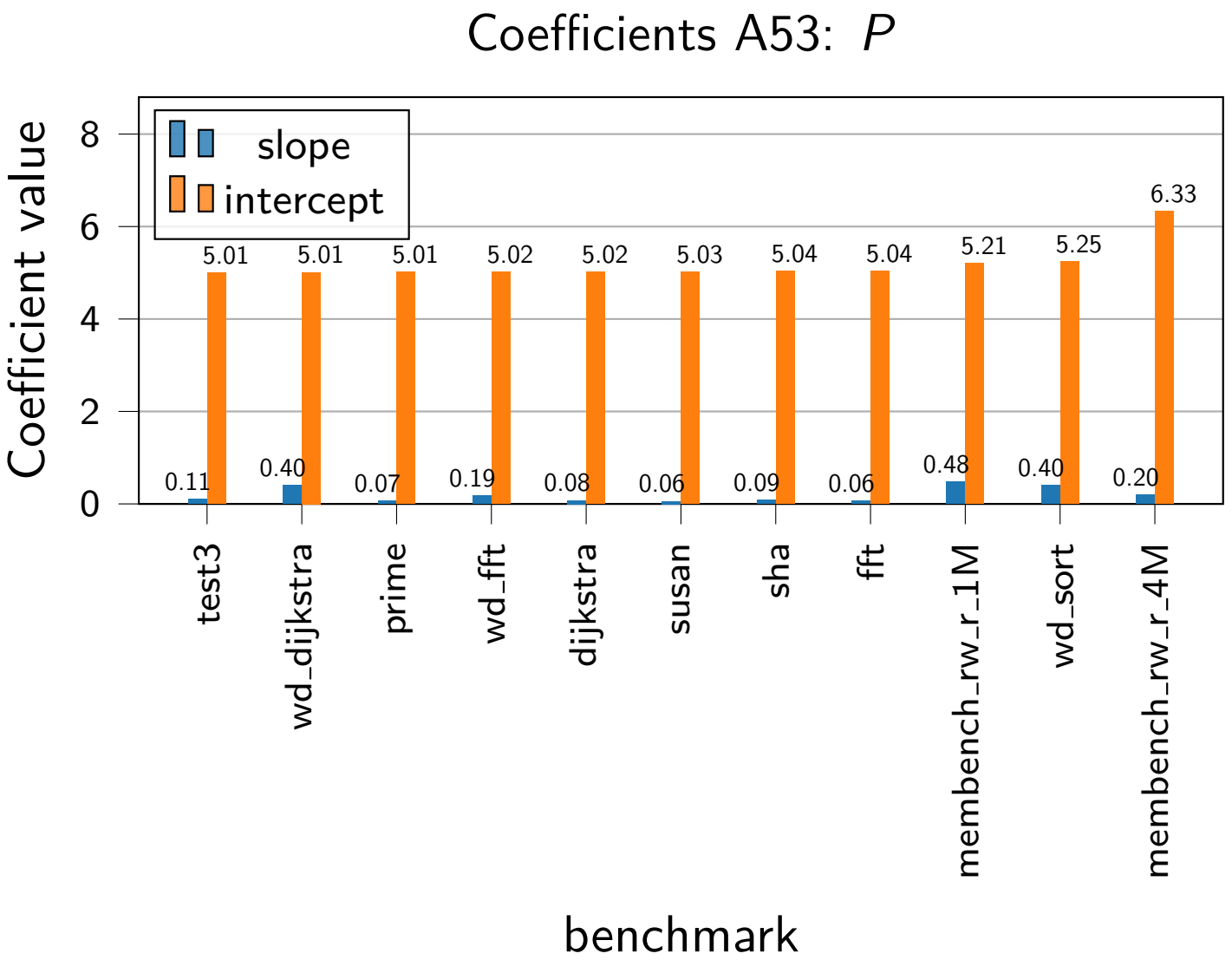
Example measurements



Identified coefficients (temperature)



Identified coefficients (power)



Combining tasks together

inside of a single window

Work-in-progress predictor

$$f(W_i, a_w, a_r) := \max_{\substack{T_i \in \mathcal{T} \\ R_k \in \mathcal{R}}} \{b_{i,k} \mid a_w(T_i) = W_i \wedge a_r(T_i) = R_k\} + \sum_{\substack{T_i \in \mathcal{T} \wedge R_k \in \mathcal{R}: \\ a_w(T_i) = W_i \\ a_r(T_i) = R_k}} a_{i,k} \quad (5)$$

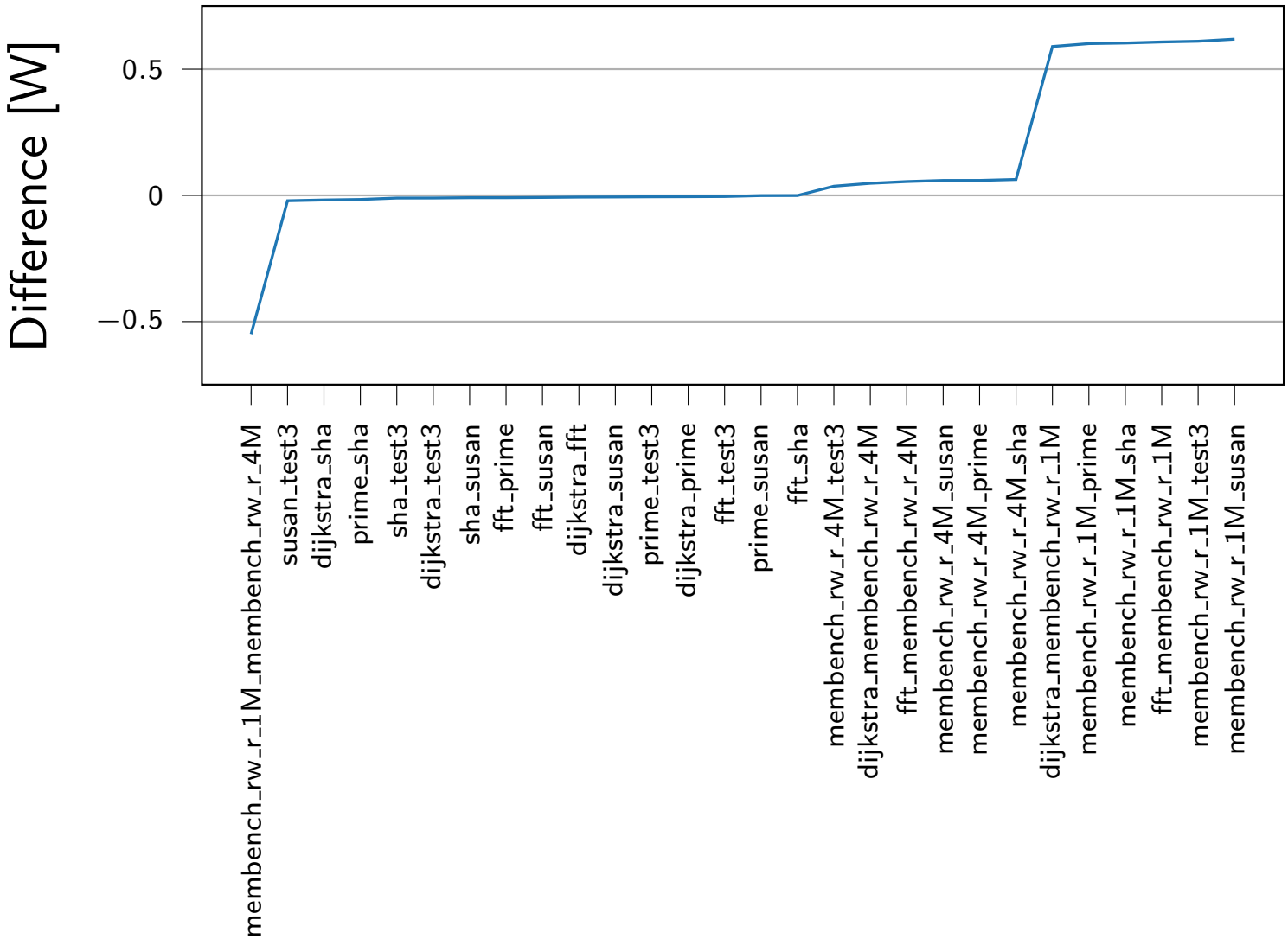
- $a_{i,k}$ – slope of task T_i for resource cluster R_k
- $b_{i,k}$ – intercept of task T_i for resource cluster R_k

'Take maximum static power among the benchmarks allocated to the given window and add the active power for all tasks allocated to the window.'

Single- and multi-core experiments follow. For single-core: 2 benchmarks were executed – one was allocated to the half of the cluster and the other one to the other half (e.g., 2 and 2 A53 cores, or 1 and 1 A72 cores). For multi-core: again 2 benchmarks were executed – each was allocated to 2x A53 + 1x A72.

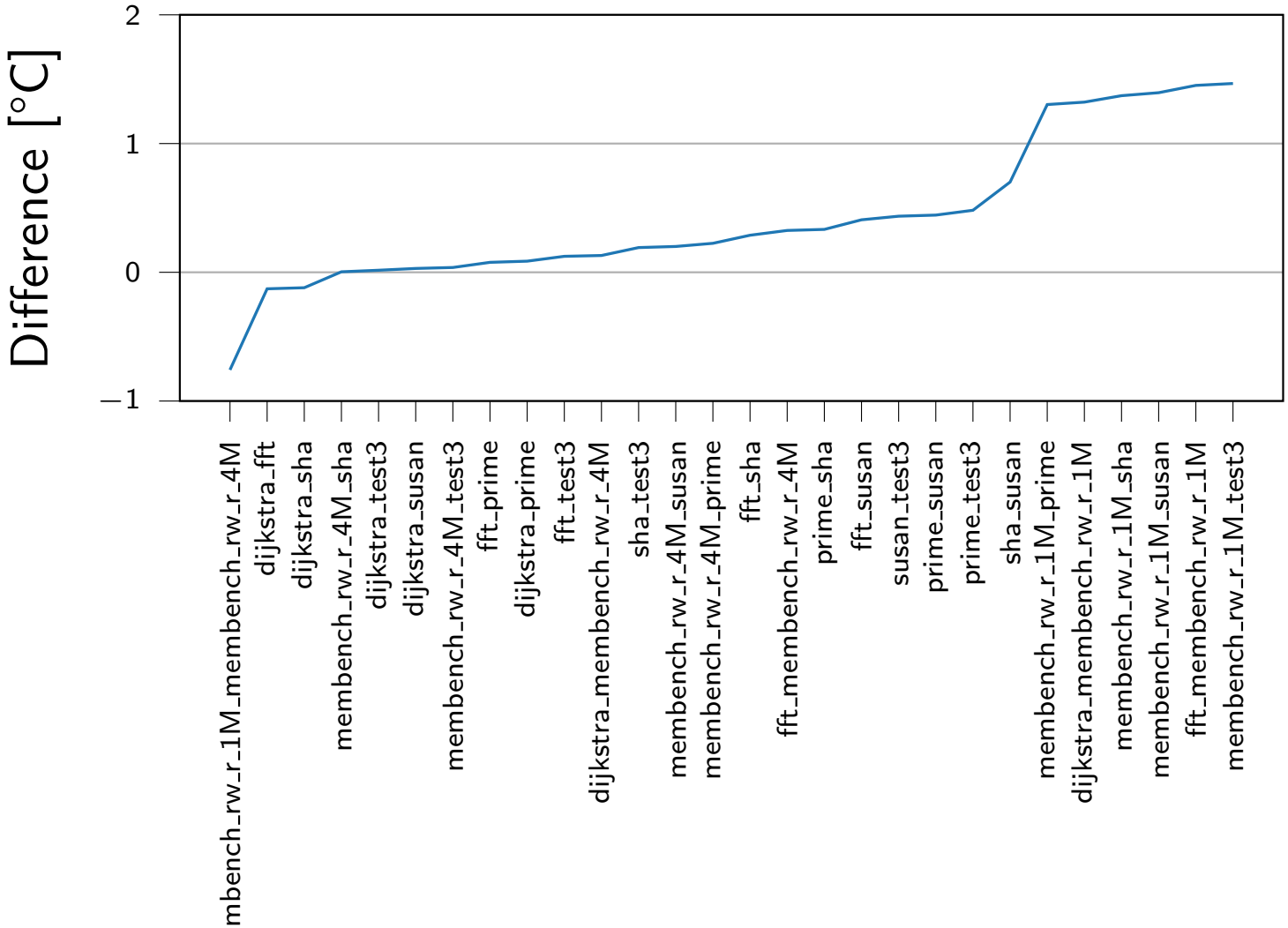
Single-cluster results (power)

Difference between measured and estimated values: A53 power



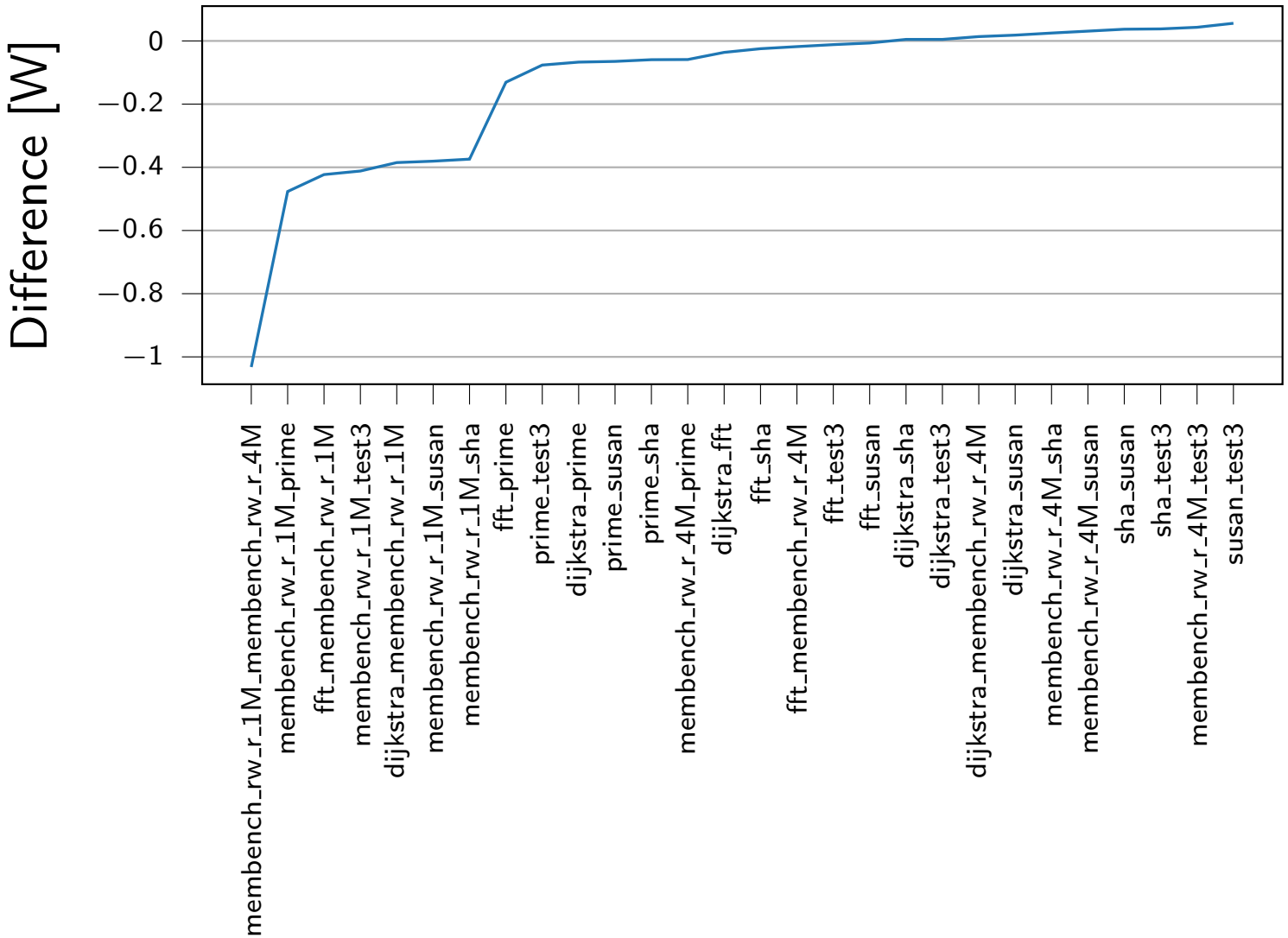
Single-cluster results (temperature)

Difference between measured and estimated values: A53 temperature



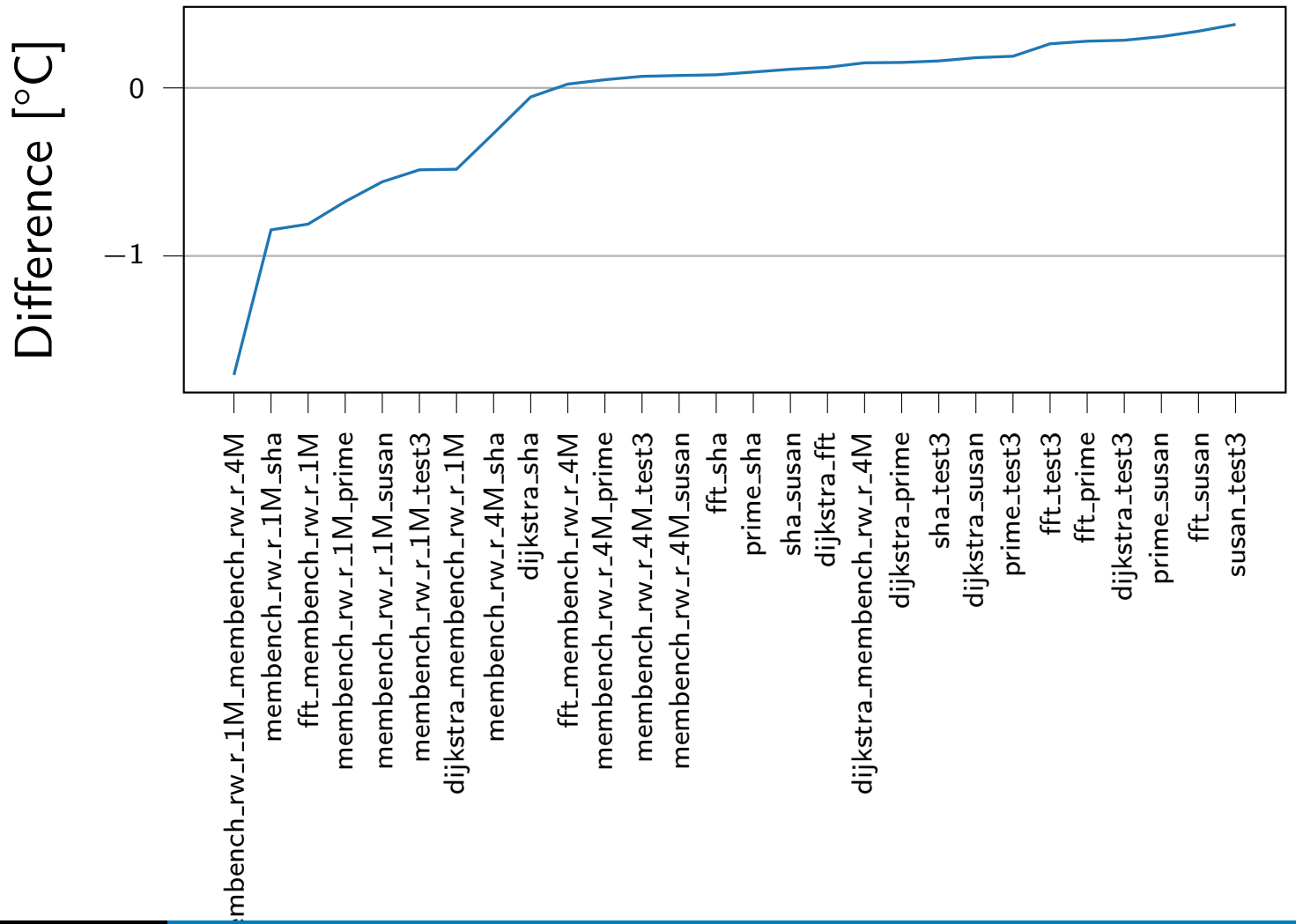
Multi-cluster results (power)

Difference between measured and estimated values: mix power



Multi-cluster results (temperature)

Difference between measured and estimated values: mix temperature



Combining windows together

Prediction for multiple windows

- take linear combinations of single windows predictions

$$f_{\text{total}}(\{W_1, \ldots, W_k\}, a_w, a_r) := \sum_{i=1}^{k} f(W_i, a_w, a_r) \cdot \frac{l_i}{h} \qquad (6)$$

Assuming that the windows are relatively short compared to the time constants of the system, the change of the temperature within a single window will not be significant. The influence of the windows' order is neglected.

Preliminary observations

Reference measurements (single core, all the time running):

Workload	T_∞ [°C]
`simd_fp32_madd`	6.5
`membench -4M`	10.0

Experiment: (repeating `mem` → `madd` → `mem` → ...)

Instance [ms]		Temperature [°C]		
`madd`	`mem`	T_{expected}	T_∞	Δ
350	50	6.9	6.8	0.1
300	100	7.4	7.1	0.3
250	150	7.8	7.8	0.0
200	200	8.3	8.1	0.2
150	250	8.7	8.7	0.0
100	300	9.1	8.9	0.2
50	350	9.6	9.6	0.0

Real experiment – to be done

- Create instances and compare optimized versus random assignments, statistically evaluate the results.