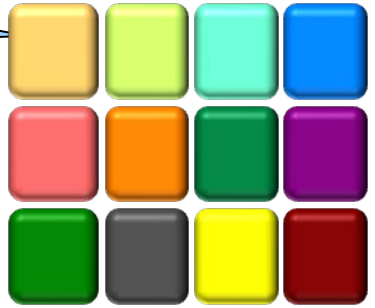
Automatic Design

Olivier TEMAM
Alchemy, INRIA Saclay, France

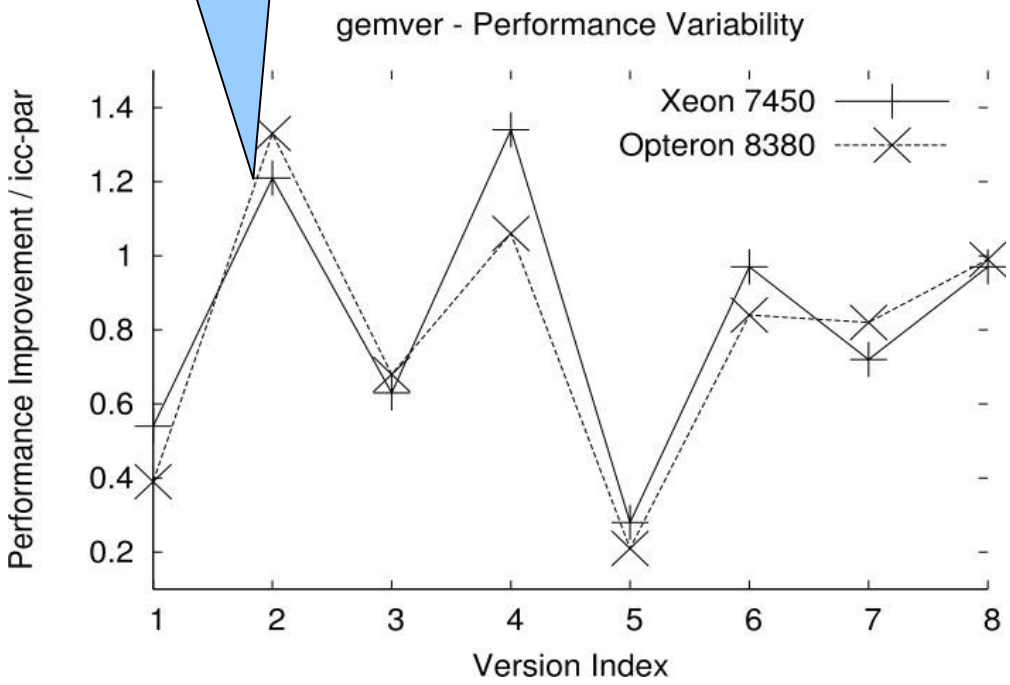
Performance Portability

- Productivity layer
  - Declarative concurrency
  - Domain-specific (algorithmic)
- Efficiency layer
  - Intermediate language for target adaptation
  - Support advanced JIT-compilation
  - For compilers, library generators and experts
  - Examples: polyhedral model extended to data-dependent control-flow, synchronous data-flow, transactional memory
- Research and transfer platforms
  - Polyhedral compilation packages
  - GCC (Graphite, Streaming, Transactional)

Variability
Heterogeneity

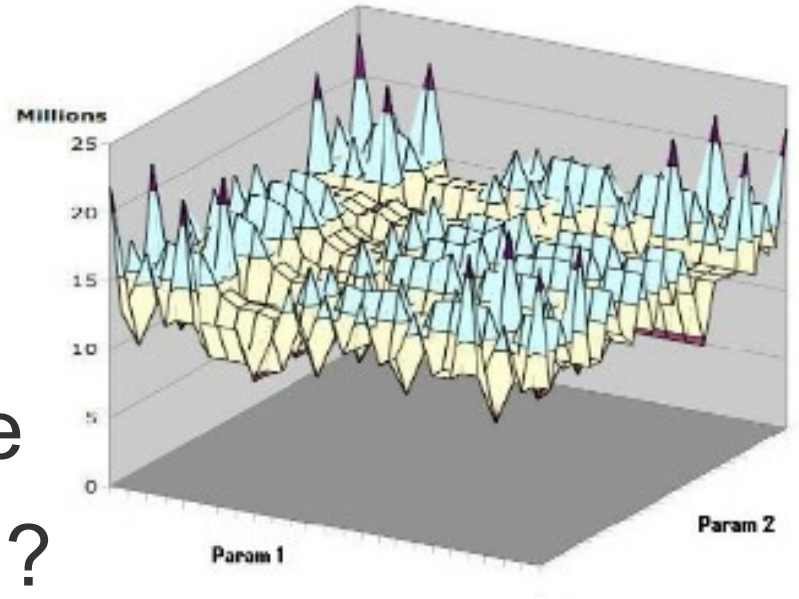


Search Space
Expressiveness,
Pruning,
Adaptivity



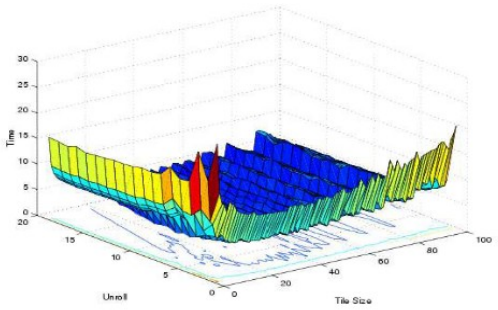
Automatic Design

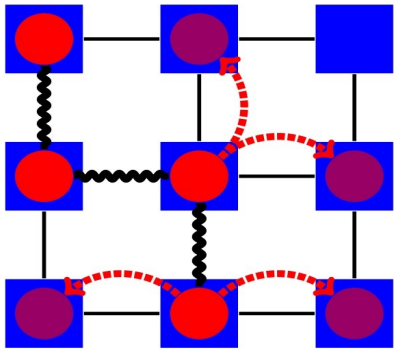
- Is there a complexity tipping point ?
- Trend towards greater complexity
- Intuition and experience still best design drivers ?

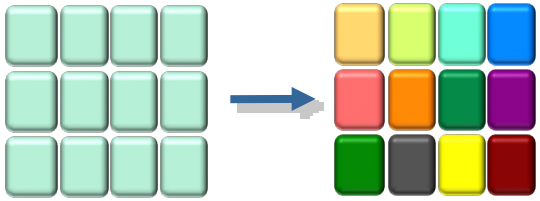


What's Needed

- Compose & Explore architectures
- Automatically re-tune compiler & programs (ctuning.org)
- Explore novel parallel architectures w/o reprogramming
- Automatically explore specialization/heterogeneous systems







An example

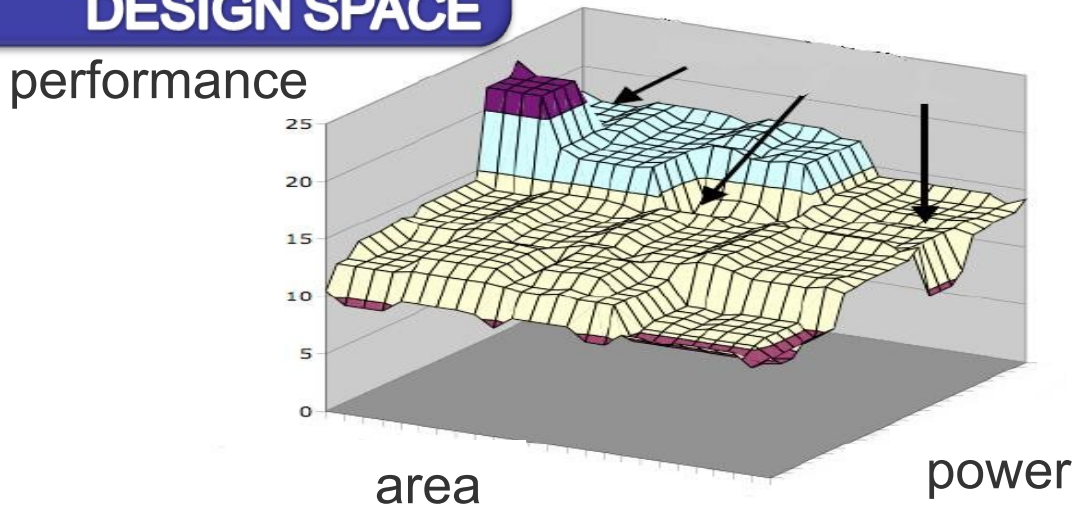
[MICRO 2004, Gracia-Pérez et al.]

Why is systematic comparison not (yet) common practice?

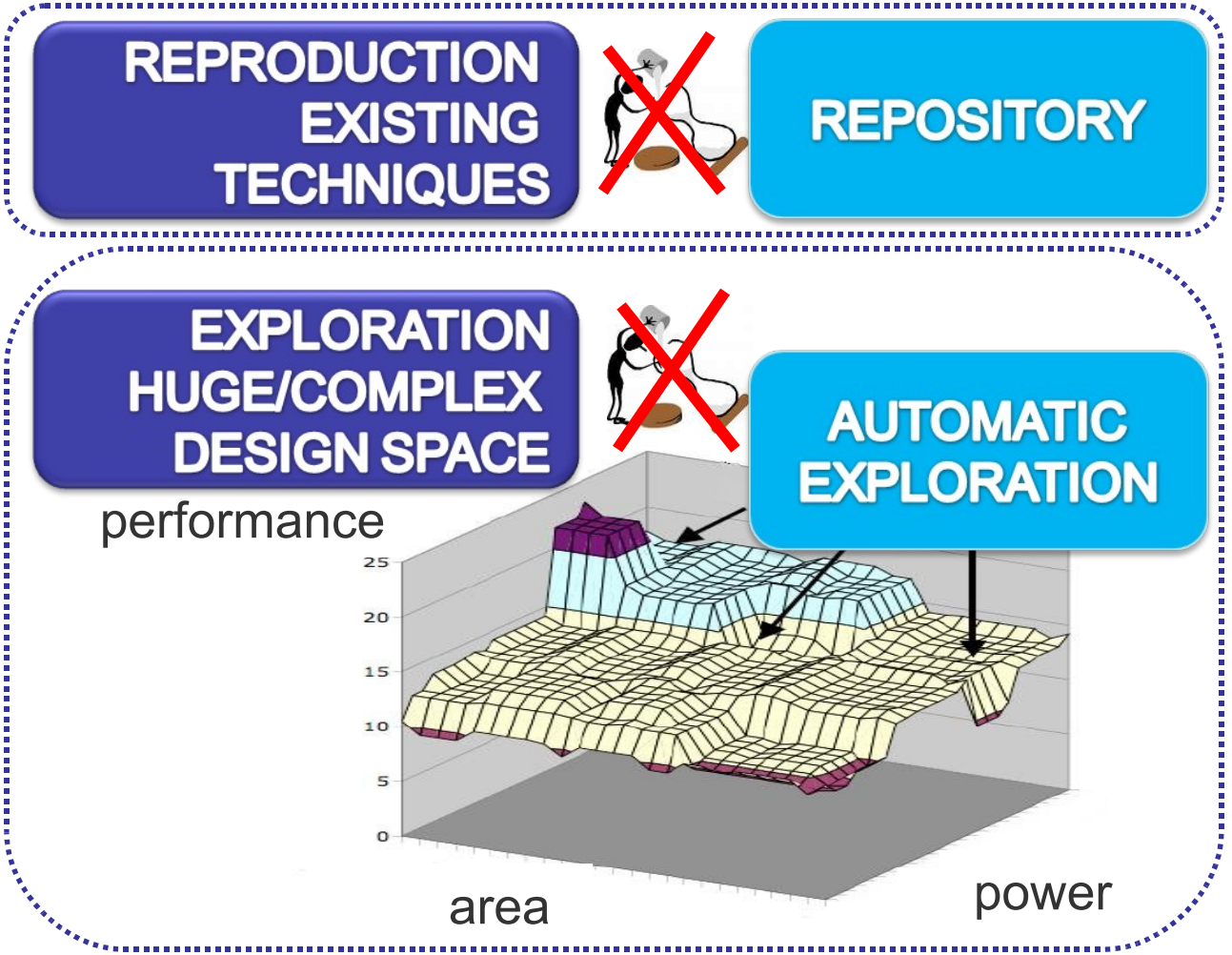
RESEARCH IDEA

REPRODUCTION EXISTING TECHNIQUES

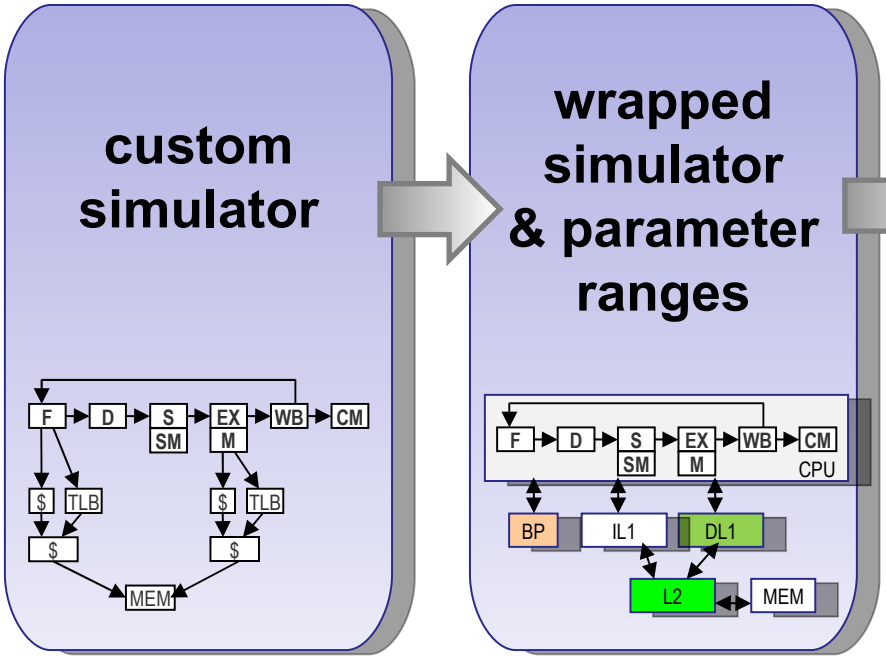
EXPLORATION HUGE/COMPLEX DESIGN SPACE



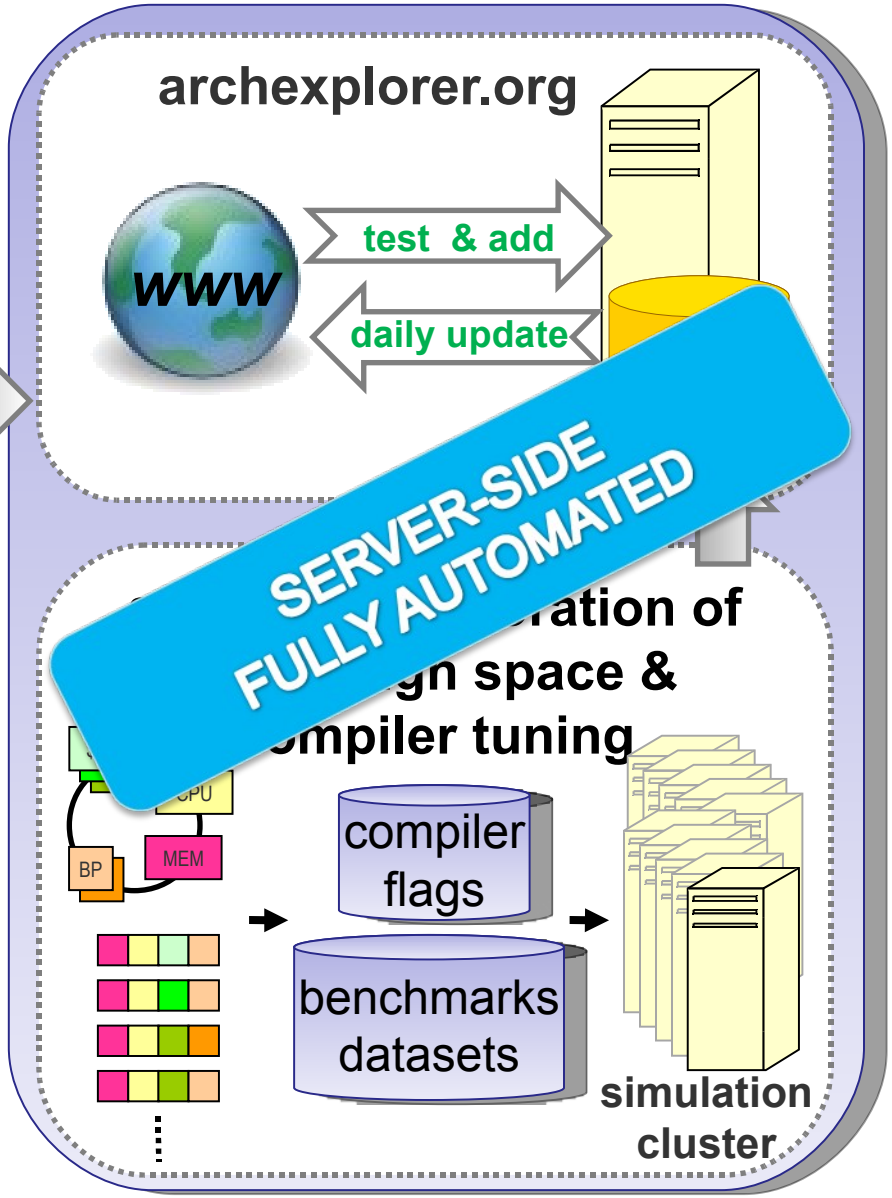
Why is systematic comparison not (yet) common practice?



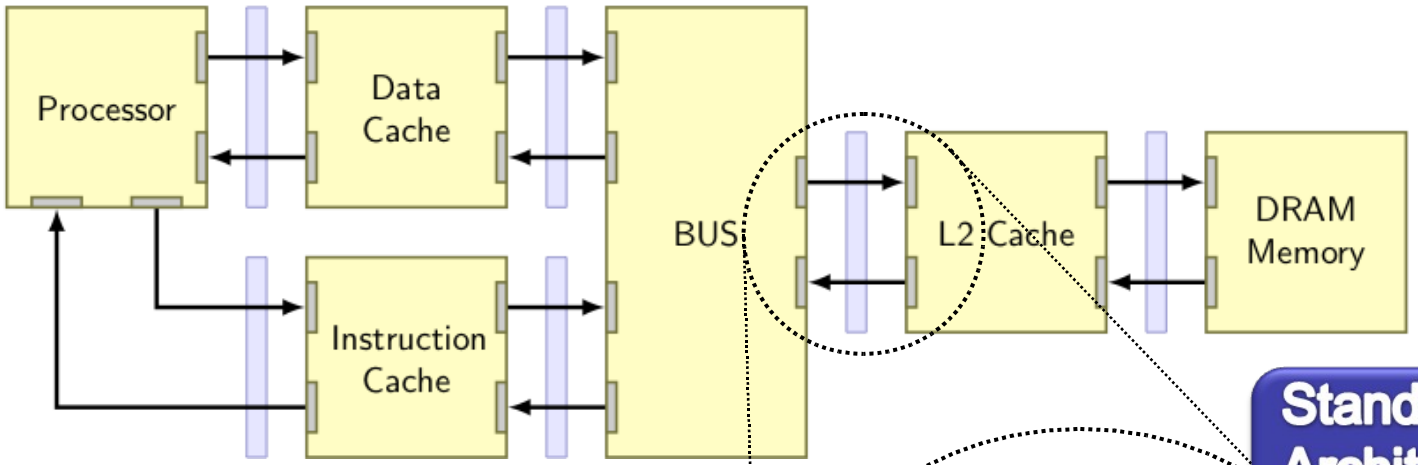
ArchExplorer



1. Hardware compatibility
2. Software compatibility
3. Parameter ranges

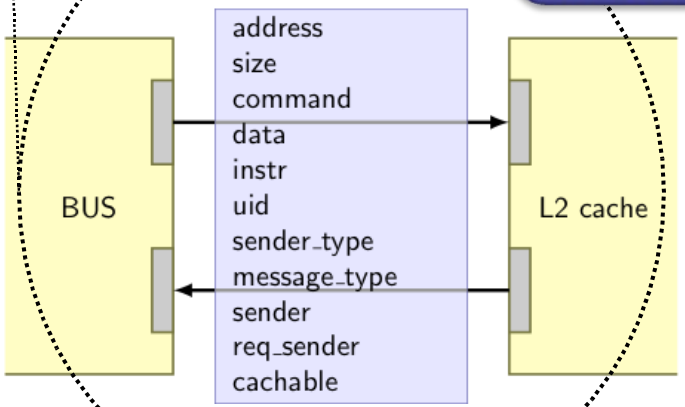


Step 1: Hardware compatibility



Caches
Branch predictors
Interconnects
Main memory
Accelerators
...

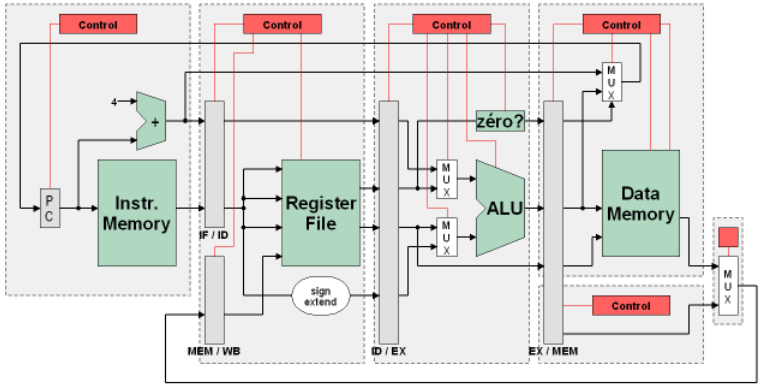
Standardize Architectural Interface



Step 2: Software compatibility

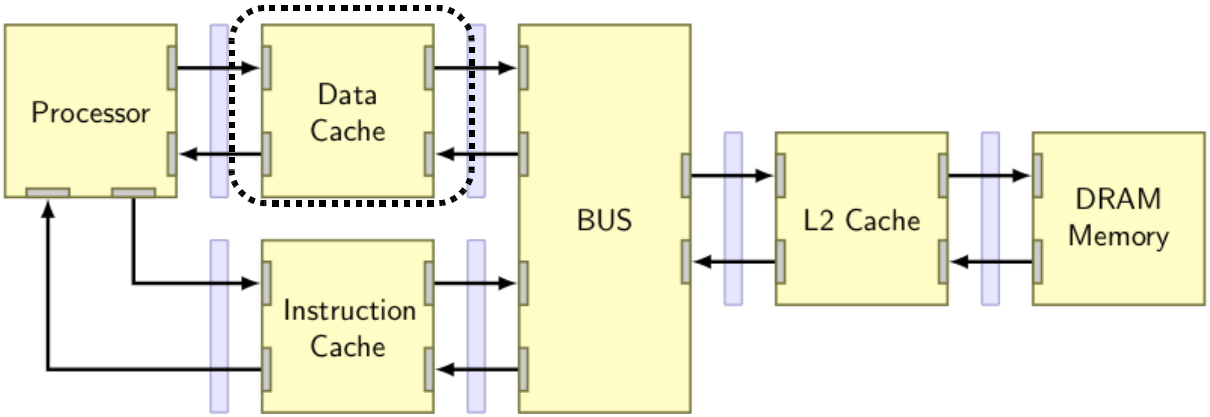
ISOLATE YOUR HARDWARE BLOCK

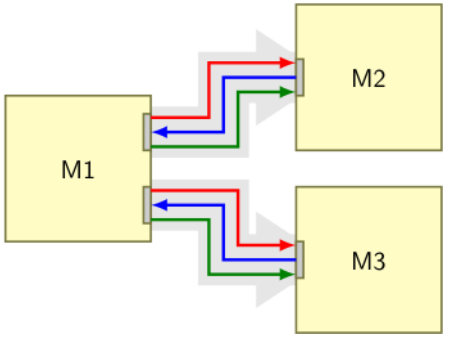
Centralized control vs. distributed control



WRAPPING IN SYSTEM-C + UNISIM

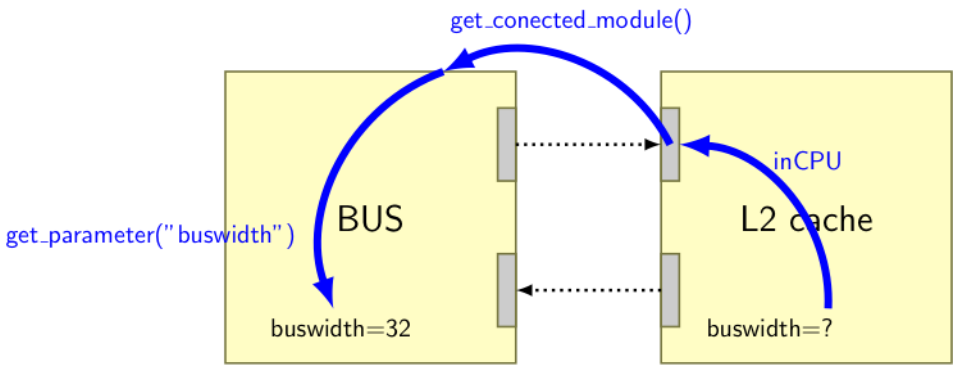
Models of computation





Step 3: Supporting Service Interfaces

Self-Configuration and parameters validity

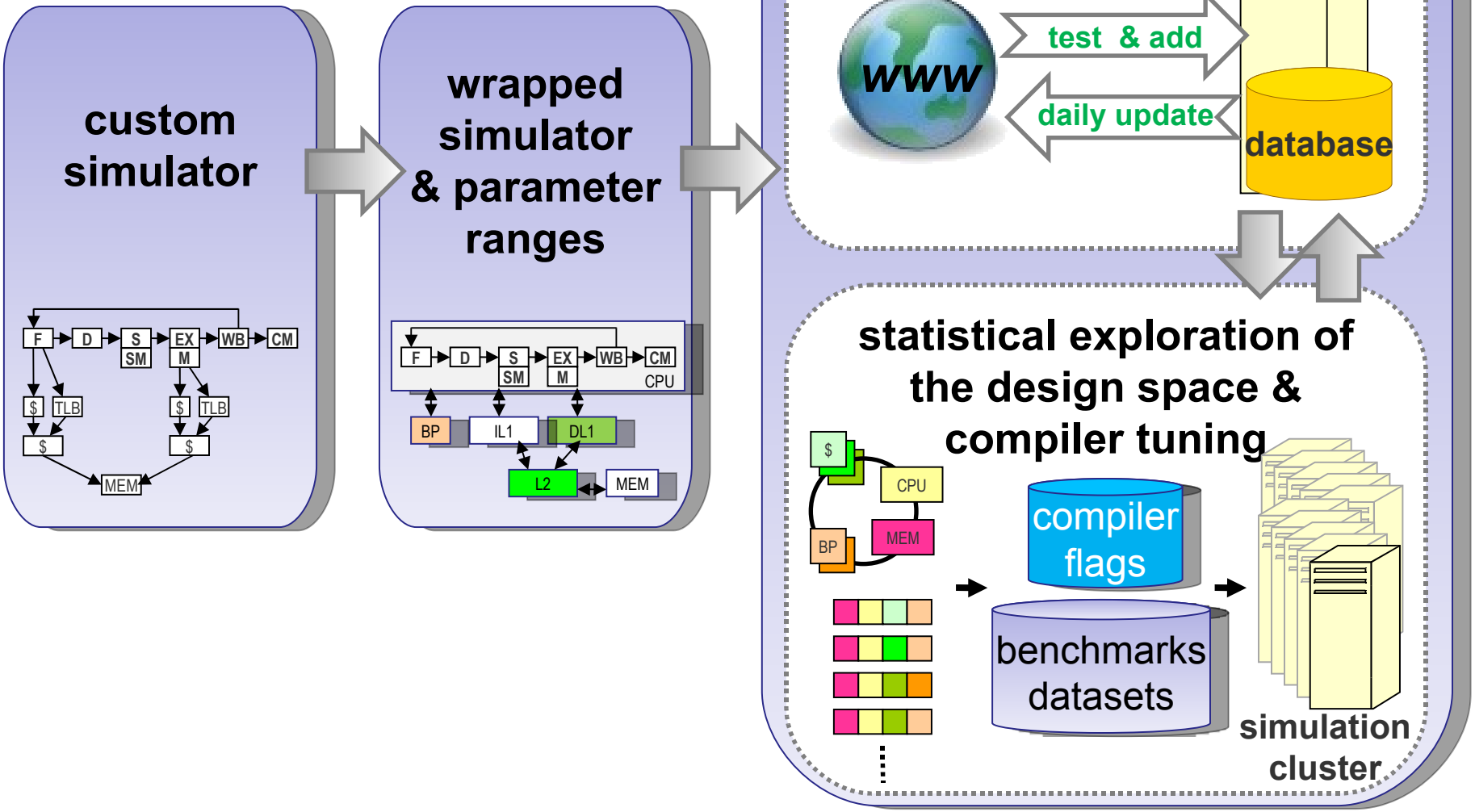


Min, max, range
Power-of-2
Positive
Complex relations

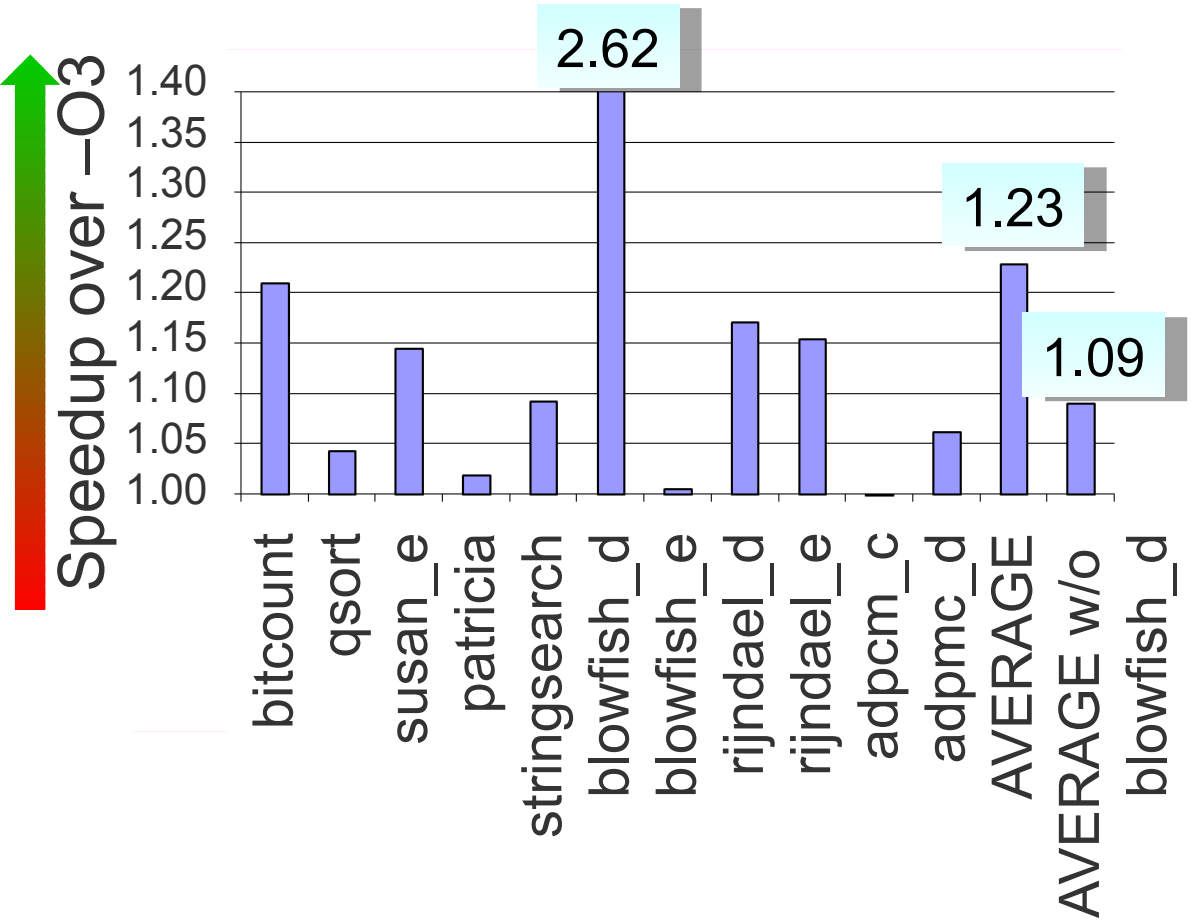
- Parameters
- Power/Cost
- Fast simulation
- Debugging
- …

Associativity 1 2 4 8
CacheLines 64 256 1024 4096
LineSize 32 64
...

ArchExplorer



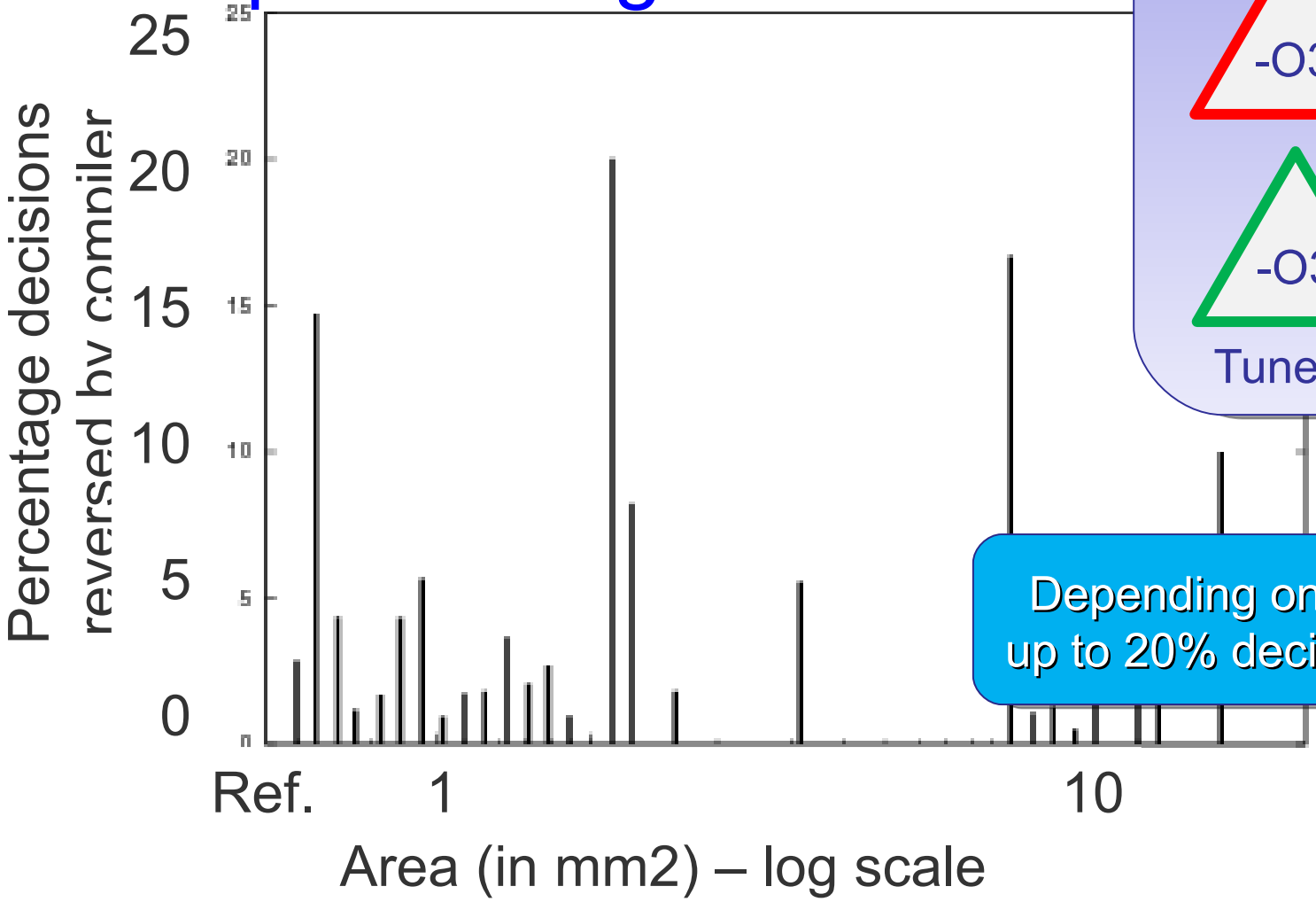
Accurate comparison needs compiler tuning as well

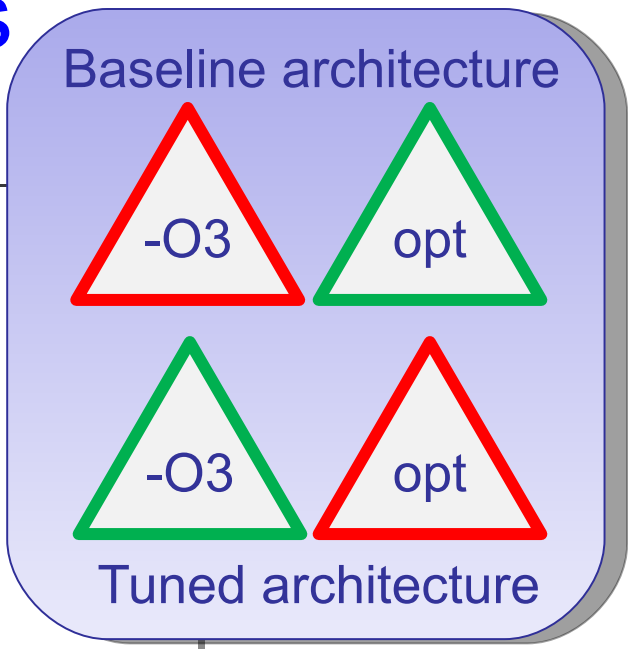


Best Compiler Flags

-O1 -falign-jumps=36 -fschedule-insns2 -fsingle-precision-constant
-O3 -falign-functions=56 -falign-jumps=51
-O1 -fno-ivopts -fno-tree-fre
-O0 -fsched-stalled-insns-dep=34 -fsched-stalled-insns=62 -falign-loops=63
-O1 -fno-rerun-cse-after-loop -fschedule-insns2
-O1 -fno-rerun-cse-after-loop -fschedule-insns2
-O1 -falign-functions=37 -falign-jumps=8 -falign-loops=40
-O1 -falign-functions=30 -falign-loops=60 –fregmove
-O1 -fsched-stalled-insns-dep=5 -fgcse-lm -finline-functions-ftree-pre
-O3 -fsched-stalled-insns-dep=54 -falign-loops=47
-O3 -fsched-stalled-insns-dep=5 -fno-tree-lrs

Accurate comparison needs compiler tuning as well

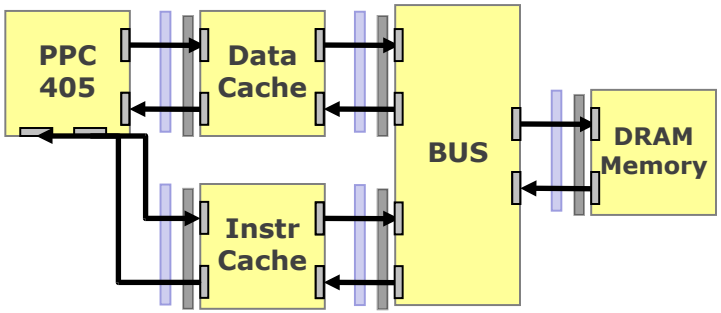




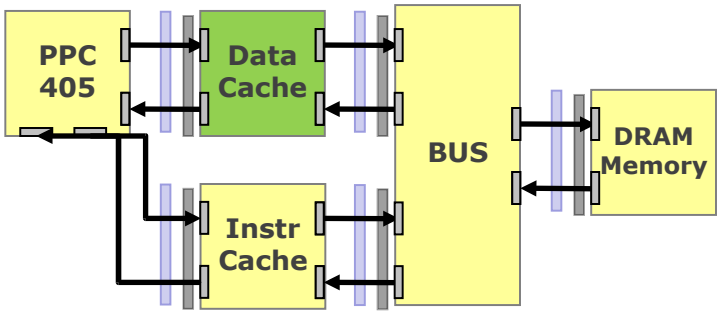
Depending on area budget up to 20% decisions reversed

Design space dimension

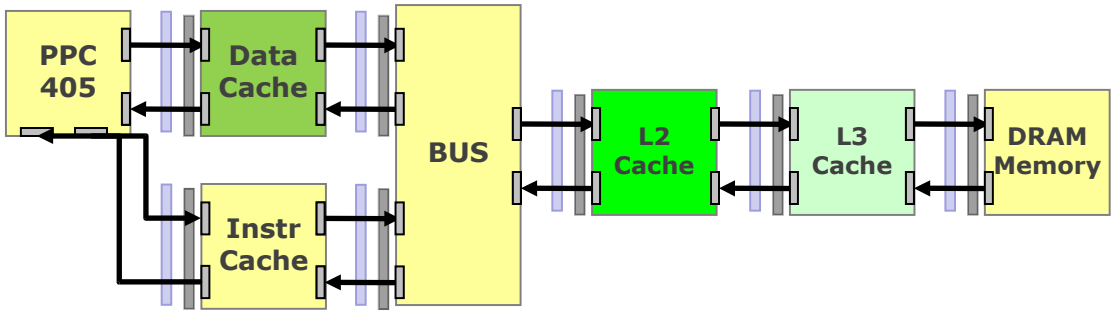
Parametric: 10^6 points



Structural: 10^9 points

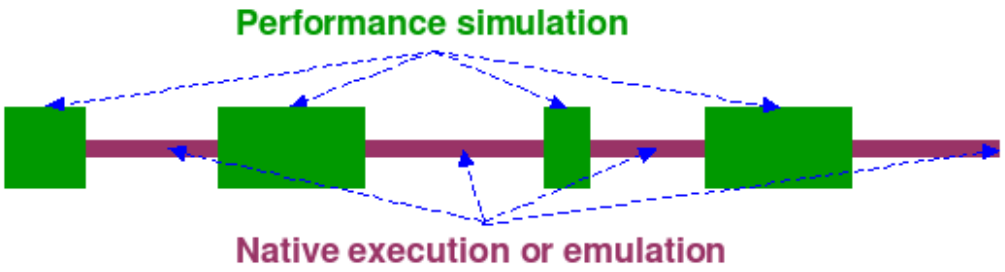


Hierarchies up to 4 levels: 10^24 points



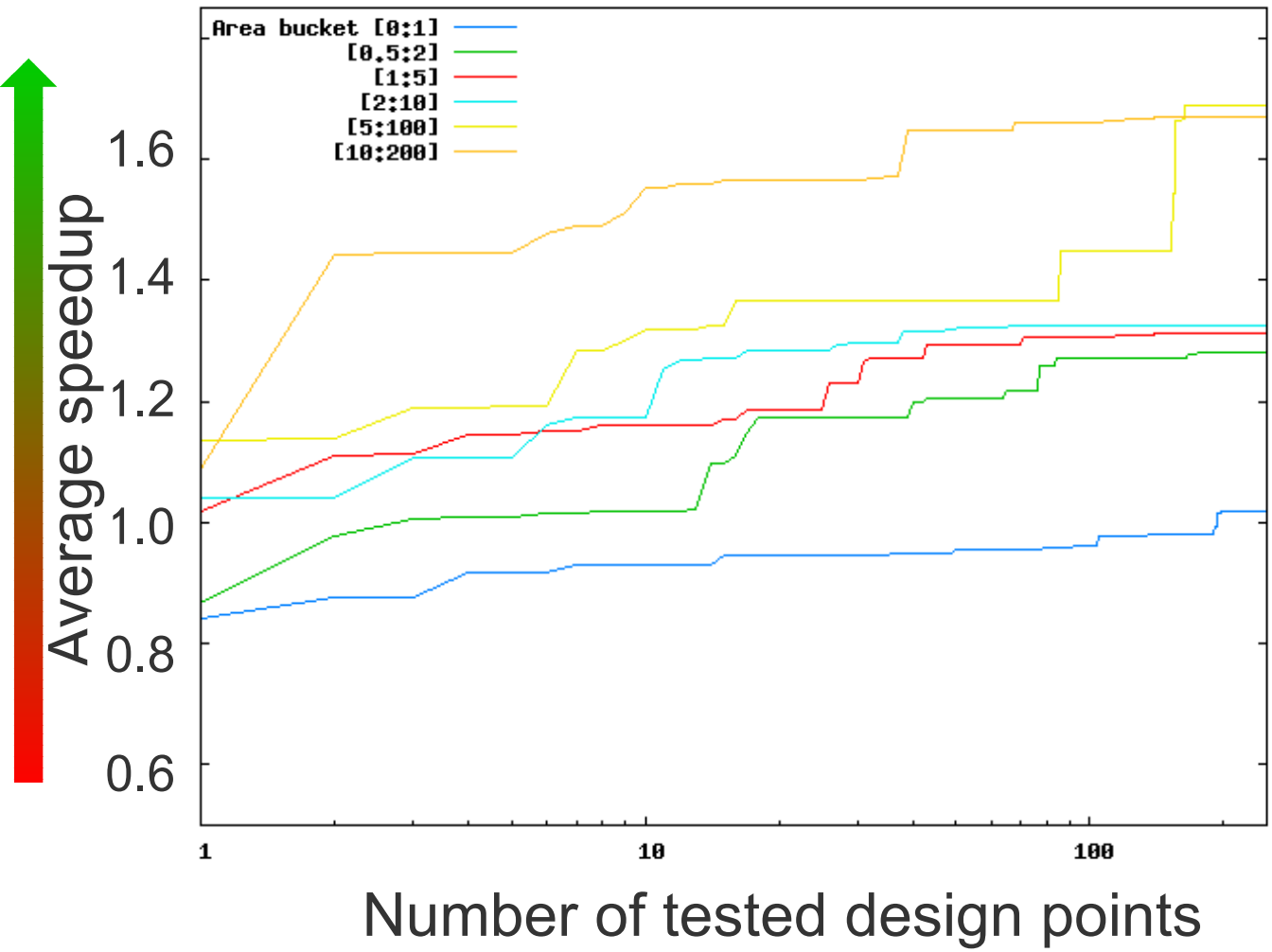
Joint architecture + compiler space: 10^27 points

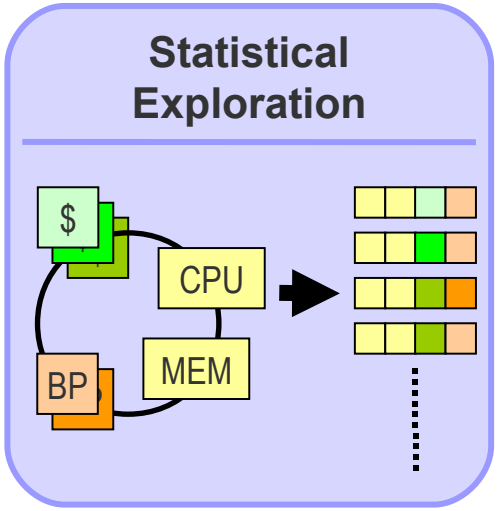
Fast evaluation



- Fast evaluation of one design point (fast simulation, e.g., sampling)
- Fast selection of design points (e.g., genetic algorithms)

Exploration scalability



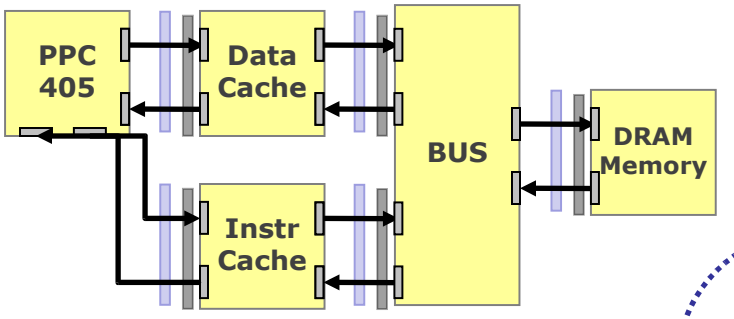


- Permanently ranks all designs
  - per area bucket
  - speedup or power
  - assigning higher probability to better points
- Picking point w.r.t distribution
- Mutations+crossover
- Natural selection

Case study

Architecture

PowerPC 405 memory sub-system



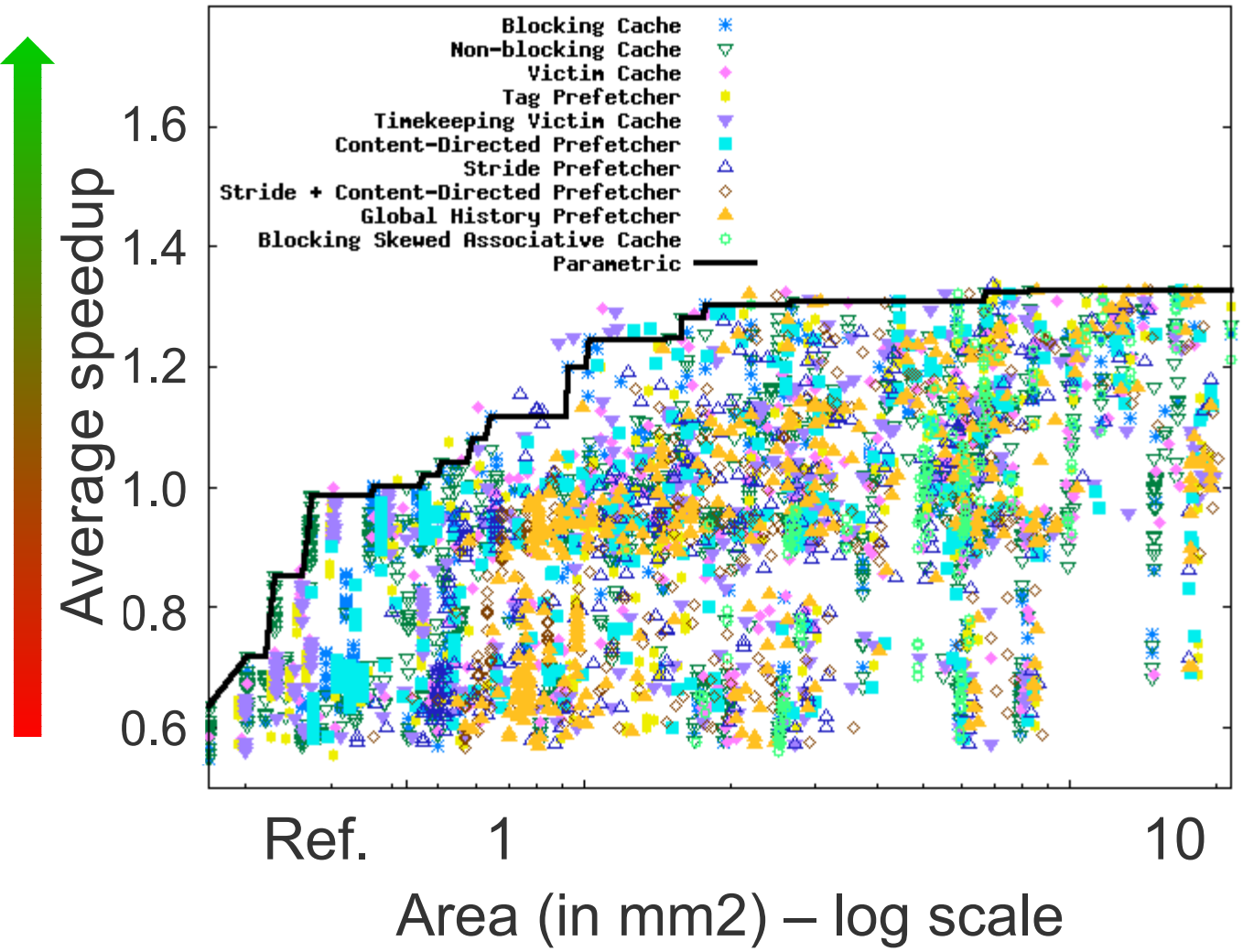
Module repository:

- Victim Cache
- Timekeeping Victim cache
- Stride Prefetcher
- Content-Directed Prefetcher
- Stride + Content Directed Prefetcher
- Tag Prefetcher
- Global History Prefetcher
- Skewed associtiative cache

Compiler

-O1 -falign-jumps=36 -fschedule-insns2 -fsingle-precision-constant
-O3 -falign-functions=56 -falign-jumps=51
-O1 -fno-ivopts -fno-tree-fre
-O0 -fsched-stalled-insns-dep=34 -fsched-stalled-insns=62 -falign-loops=63
-O1 -fno-rerun-cse-after-loop -fschedule-insns2
-O1 -fno-rerun-cse-after-loop -fschedule-insns2
-O1 -falign-functions=37 -falign-jumps=8 -falign-loops=40
-O1 -falign-functions=30 -falign-loops=60 –fregmove
-O1 -fsched-stalled-insns-dep=5 -fgcse-lm -finline-functions-ftree-pre
-O3 -fsched-stalled-insns-dep=54 -falign-loops=47
-O3 -fsched-stalled-insns-dep=5 -fno-tree-lrs

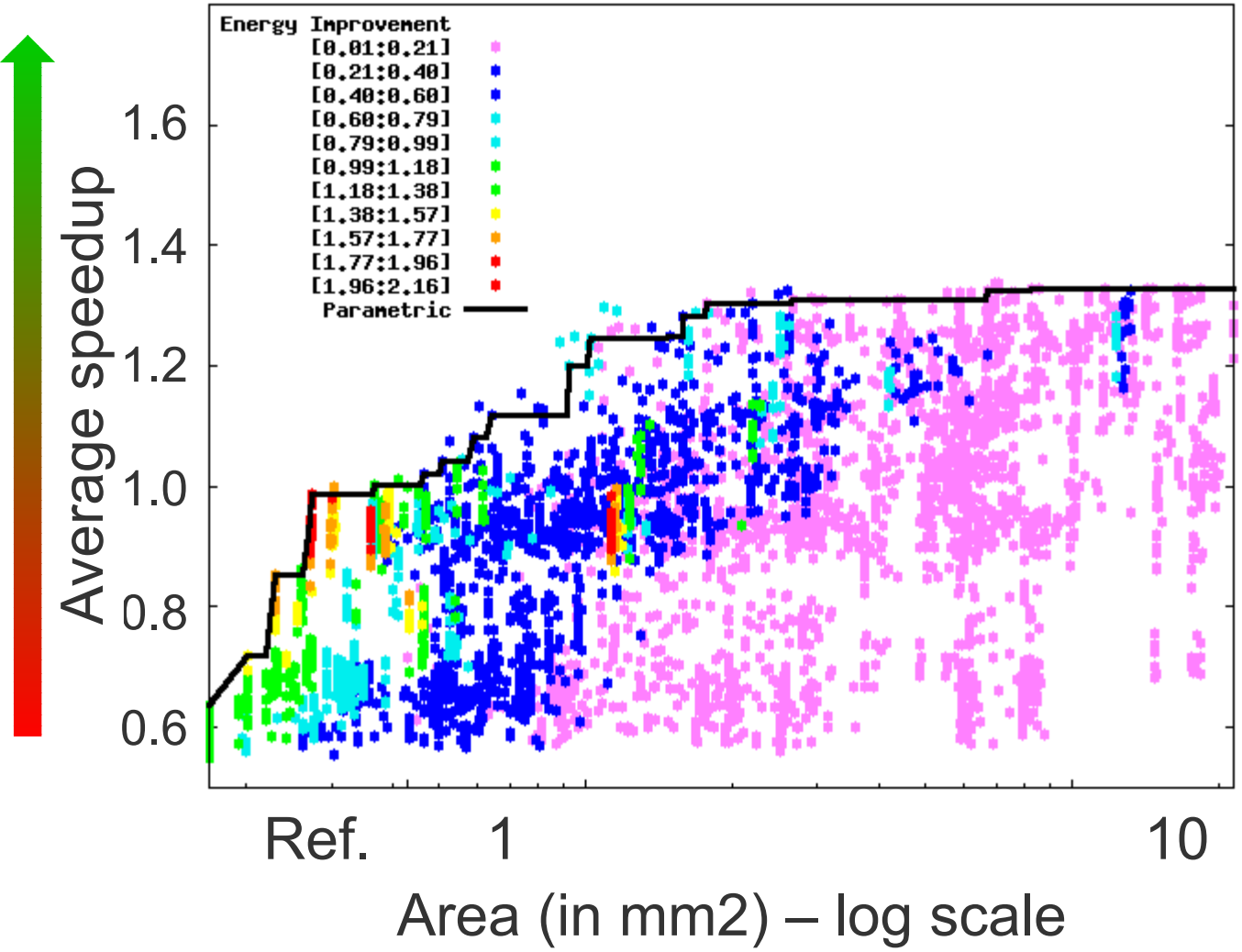
Best memory sub-system per area



CONCLUSIONS:

1. Contrast to Gracia-Pérez et al. [MICRO 2004]
2. No clear winner
3. Close to tuned parametric cache

Speedup and energy improvement



Check out this website:

ARCHEXPLORER.ORG

Permanent on-line competition - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://archexplorer.org/ranking.html

Google Agenda Cambridge

Google Agenda | Permanent on-line competition | Gmail - Inbox - veerle.desmet@g... | E-mail :: Inbox

Design Space Exploration

Permanent on-line competition(s)

Home | Ranking | Download | HowTo | Participate

Last updated: Tue Nov 25 22:08:12 CET 2008

permanently updated

Participate and evaluate your own mechanisms

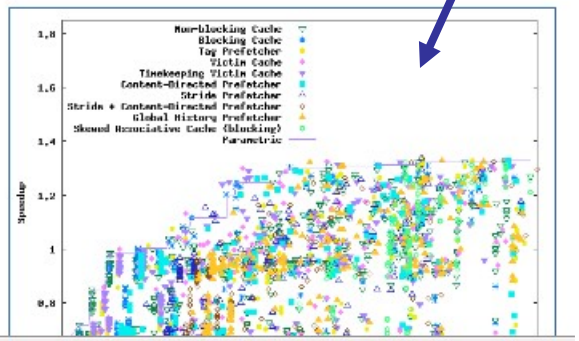
Ranking of available cache mechanisms

Only top points, i.e. points that are above the parametric envelope either performance-wise or energy-wise, are listed. To sort, click on column title.

details for top points

Cache Mechanism	Area in mm2 (memory subsystem only)	Speedup (over PowerPC405)	Energy Improvement (over PowerPC405)
Stride Prefetcher [8]	6.938	1.339	0.046
Tag Prefetcher [7]	7.050	1.339	0.047
Tag Prefetcher [7]	11.714	1.328	0.046
Global History Prefetcher [9]	8.568	1.327	0.046
Stride + Content-Directed Prefetcher [12]	14.609	1.327	0.148
Victim Cache	2.601	1.326	0.572

all design points

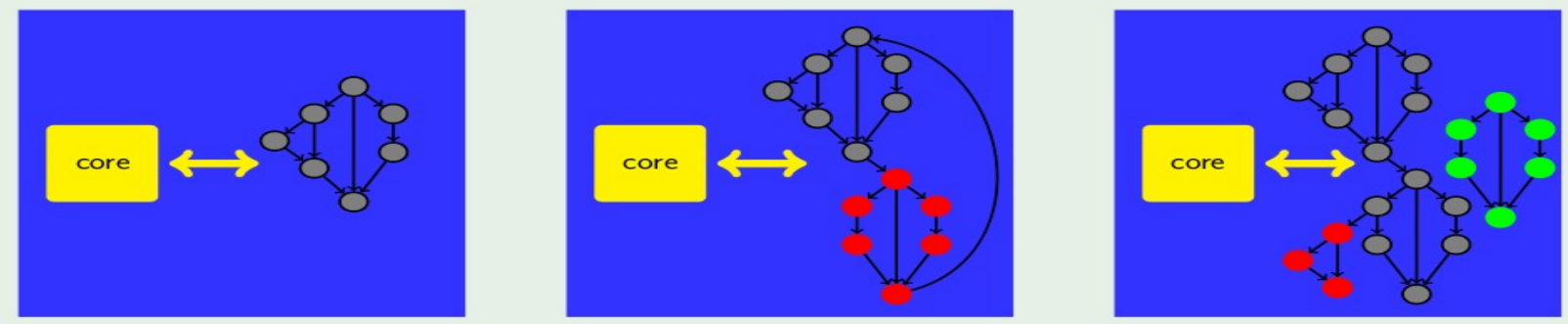


http://archexplorer.org/fig_parametric_structural.pdf

Conclusions & future work

- ArchExplorer.org
  - Facilitates fair quantitative comparison of research ideas
    - repository
    - automatic, joint compiler/hardware design space exploration
- Permanent open competition(s)
  - memory sub-system competition
- Future work:
  - explorable customization

Next step: explorable customization



- Customization “on-demand”
- Automatically generate & aggregate accelerators