ASPLOS XXI - Atlanta, Georgia – 4 April 2016

WHIRLPOOL!

IMPROVING DYNAMIC CACHE MANAGEMENT WITH STATIC DATA CLASSIFICATION

Anurag Mukkara, *Nathan Beckmann*, Daniel Sanchez

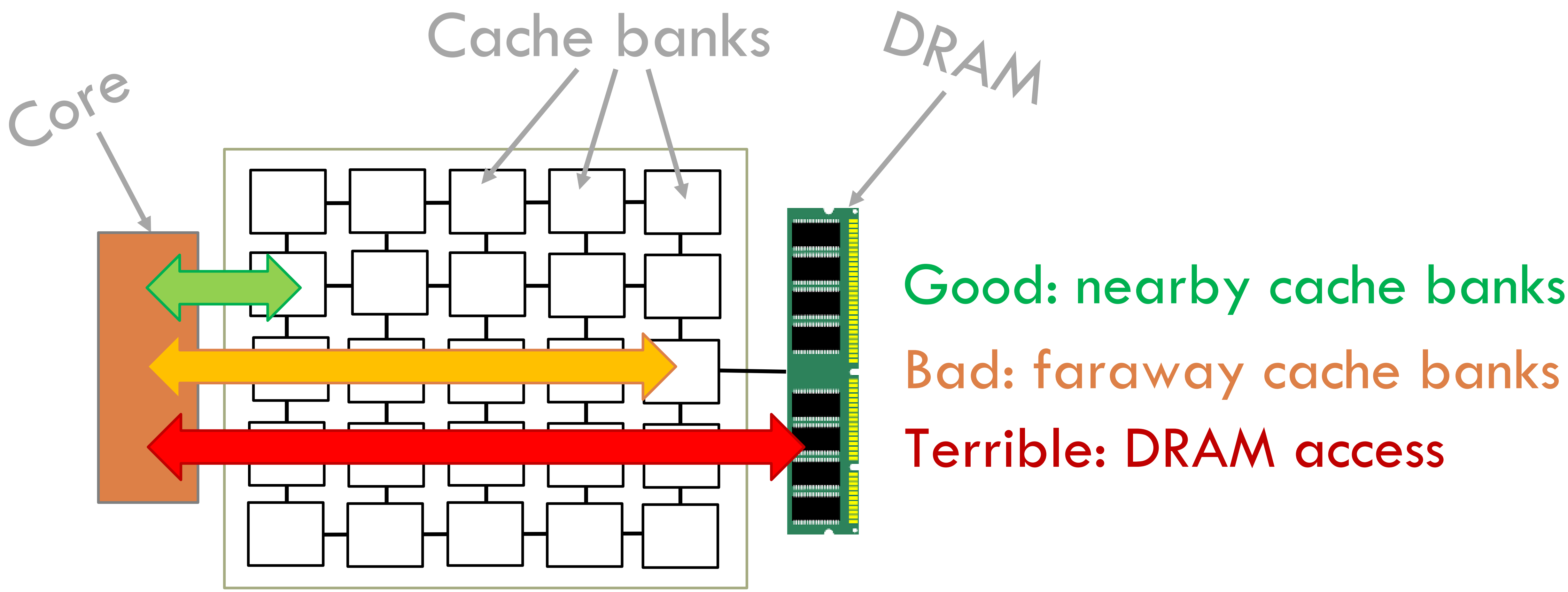
MIT CSAIL

Massachusetts Institute of Technology

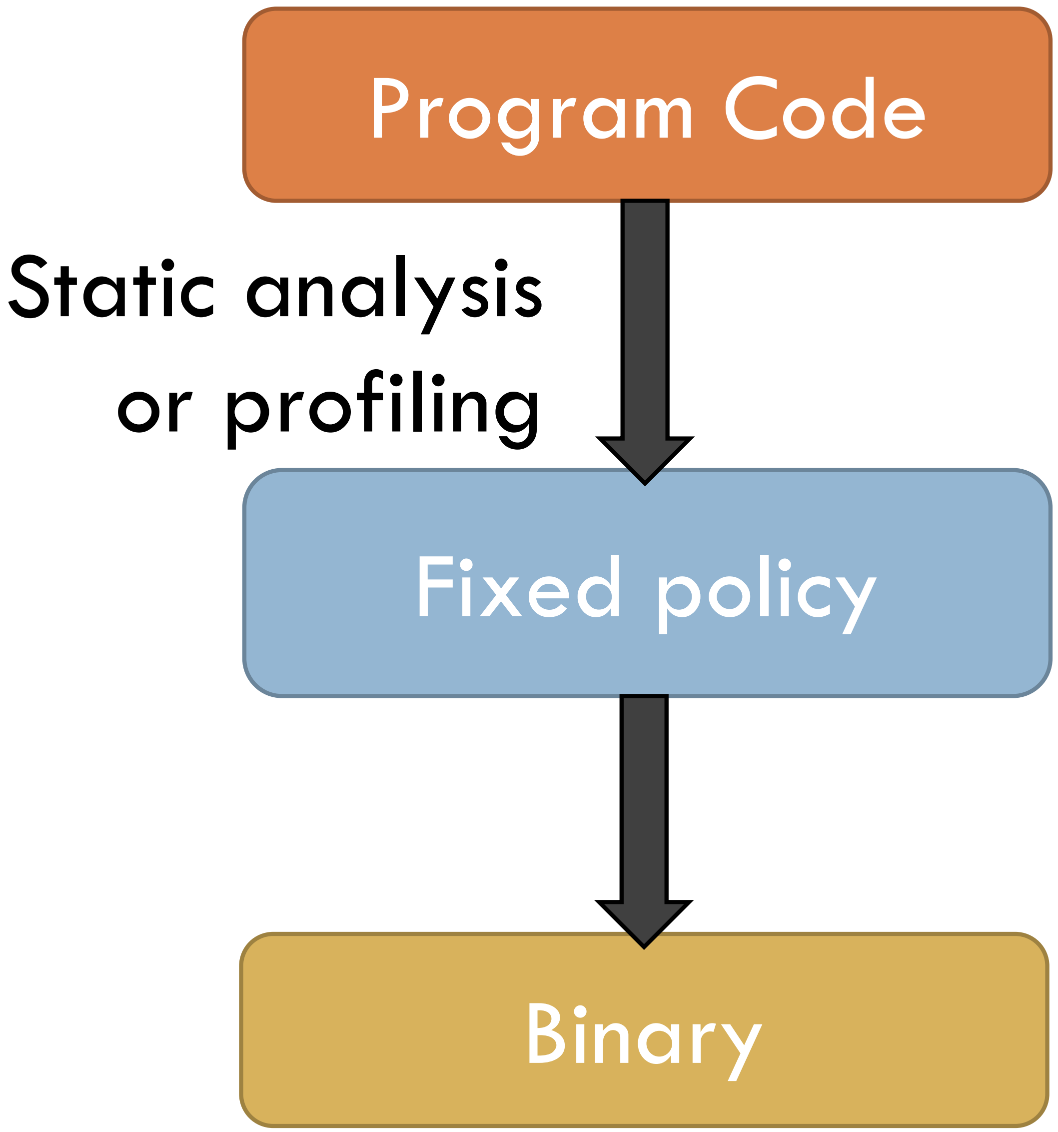
CSAIL

Processors are limited by data movement

- Data movement often consumes >50% of time & energy
  - E.g., FP multiply-add: 20 pJ ⇔ DRAM access: 20,000 pJ
- To scale performance, must keep data near where its used
- *But how do programs use memory?*



Static policies have limitations

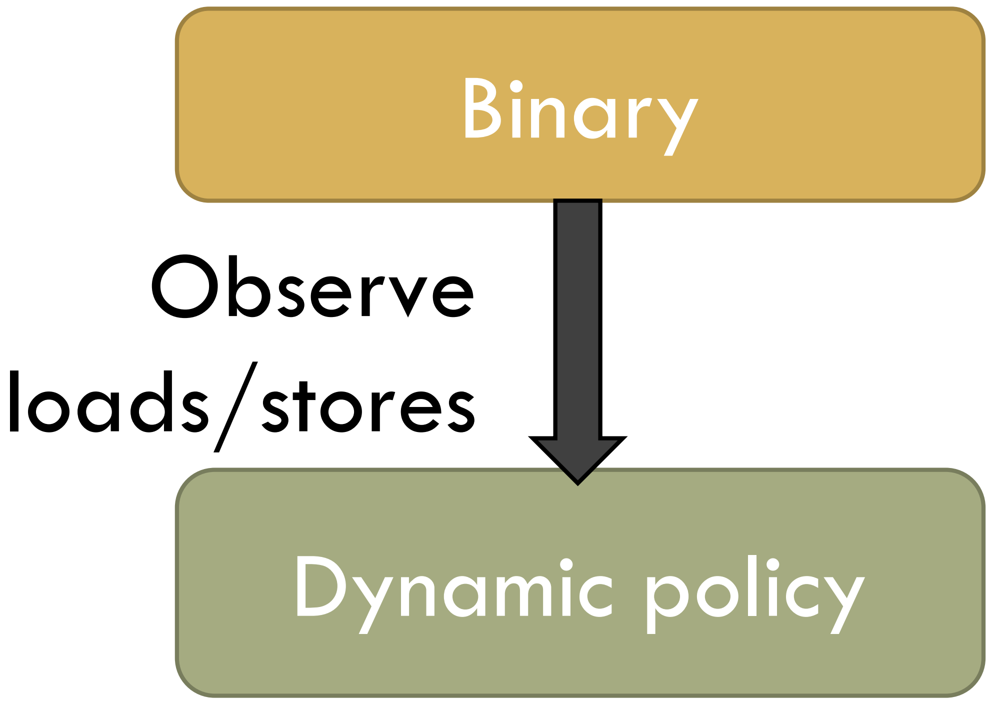


Exploits program semantics

Can’t adapt to application phases, input-dependent behavior, or shared systems

E.g., scratchpads, bypass hints

Dynamic policies have limitations, too



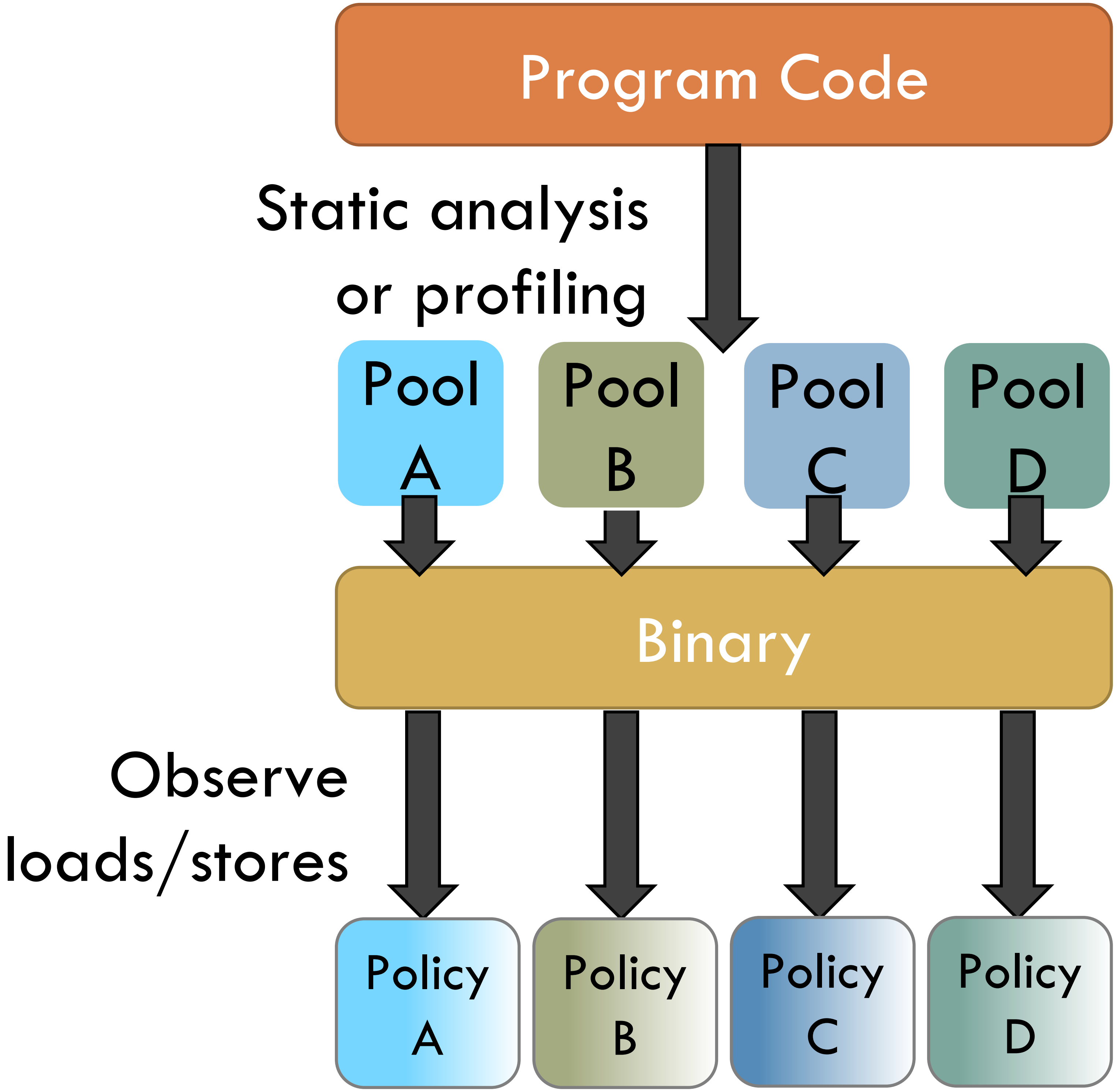
E.g., data migration & replication

Responsive to actual application behavior

Difficult to recover program semantics from loads/stores

→ Expensive mechanisms (eg, extra data movement & directories)

Combining static and dynamic is best



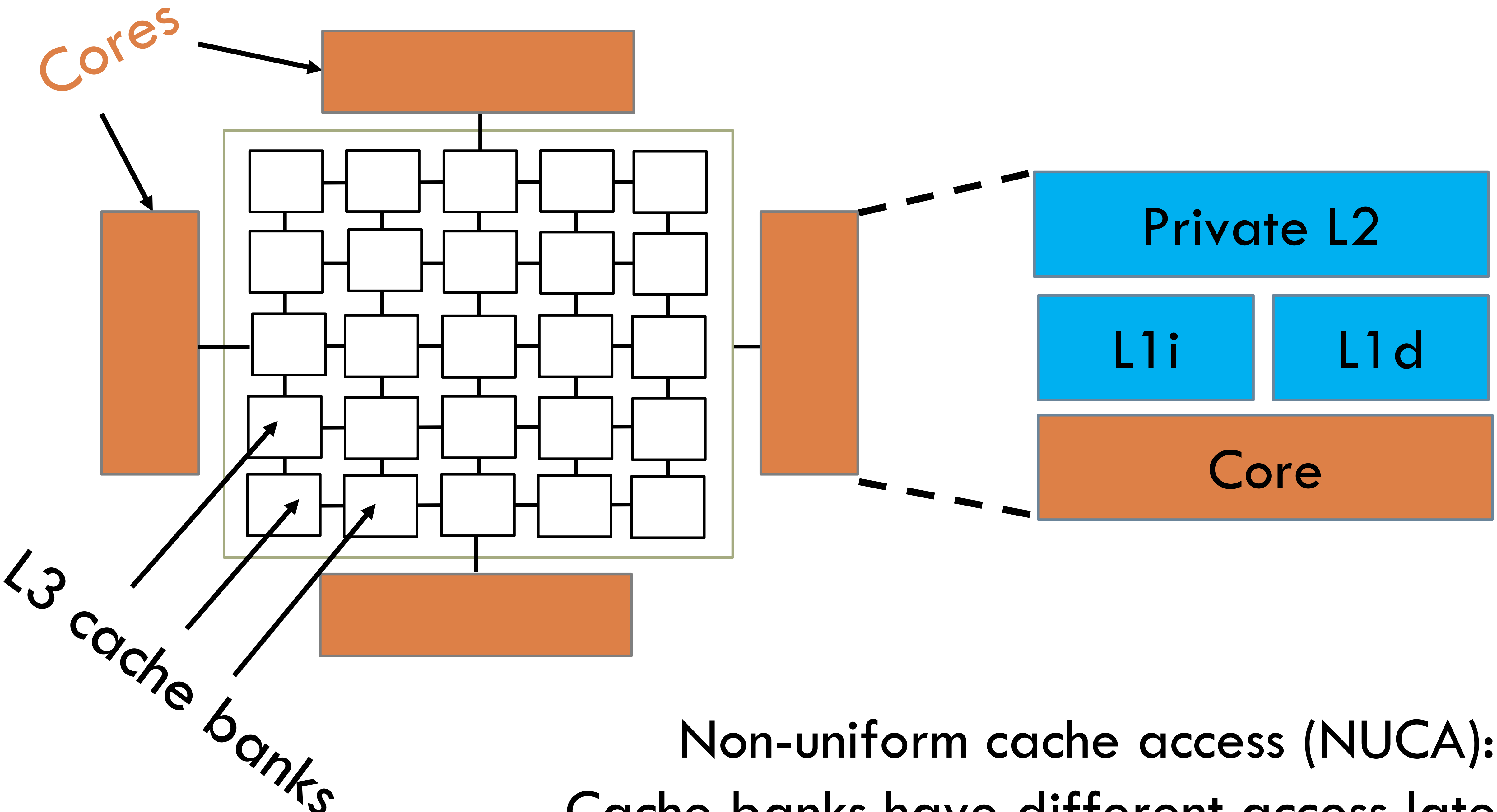
Exploits program semantics at low overhead

Responsive to actual application behavior

Agenda

- Case study
- Manual classification
- Parallel applications
- WhirlTool

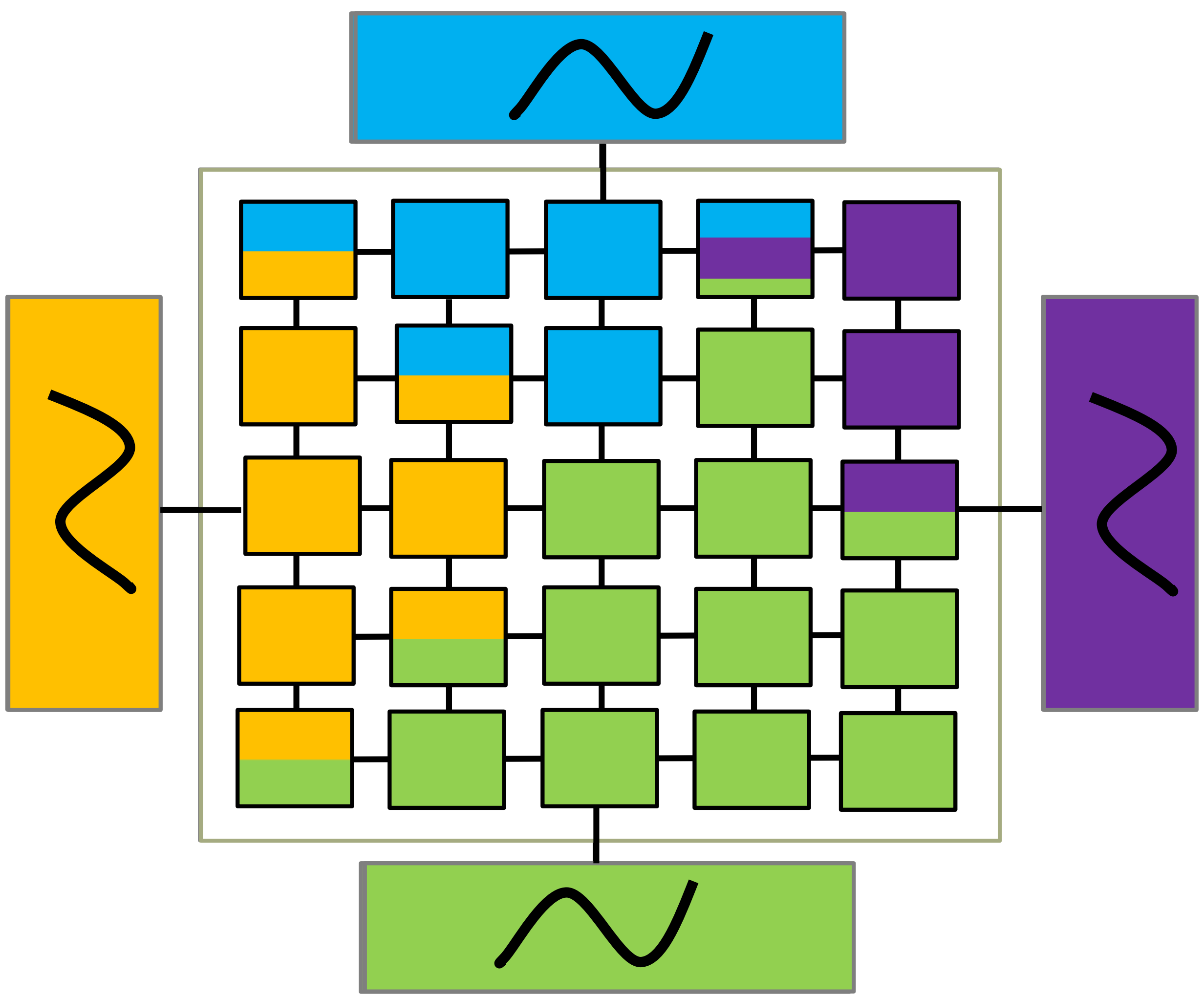
System configuration



Non-uniform cache access (NUCA):
Cache banks have different access latencies

Baseline dynamic NUCA scheme

- We apply Whirlpool to **Jigsaw** *[Beckmann PACT'13]*, a state-of-the-art NUCA cache
  - Allocates *virtual caches*, collections of parts of cache banks
  - Significantly outperforms prior D-NUCA schemes



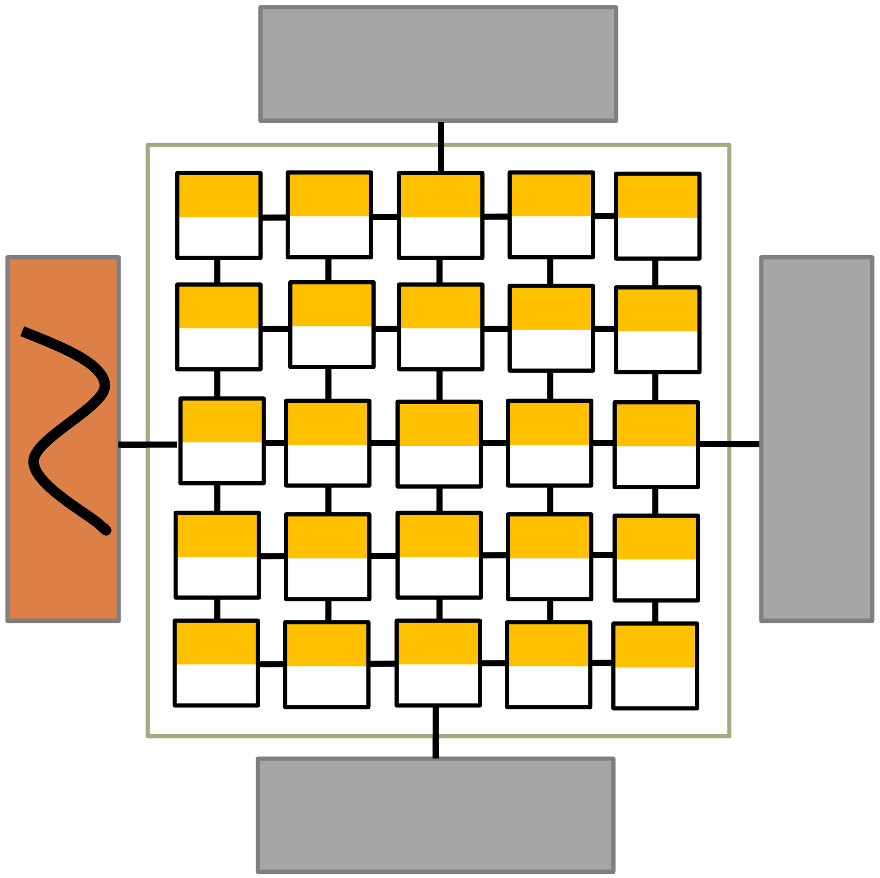
✓ Reduce cache misses

✓ Reduce on-chip network traversals

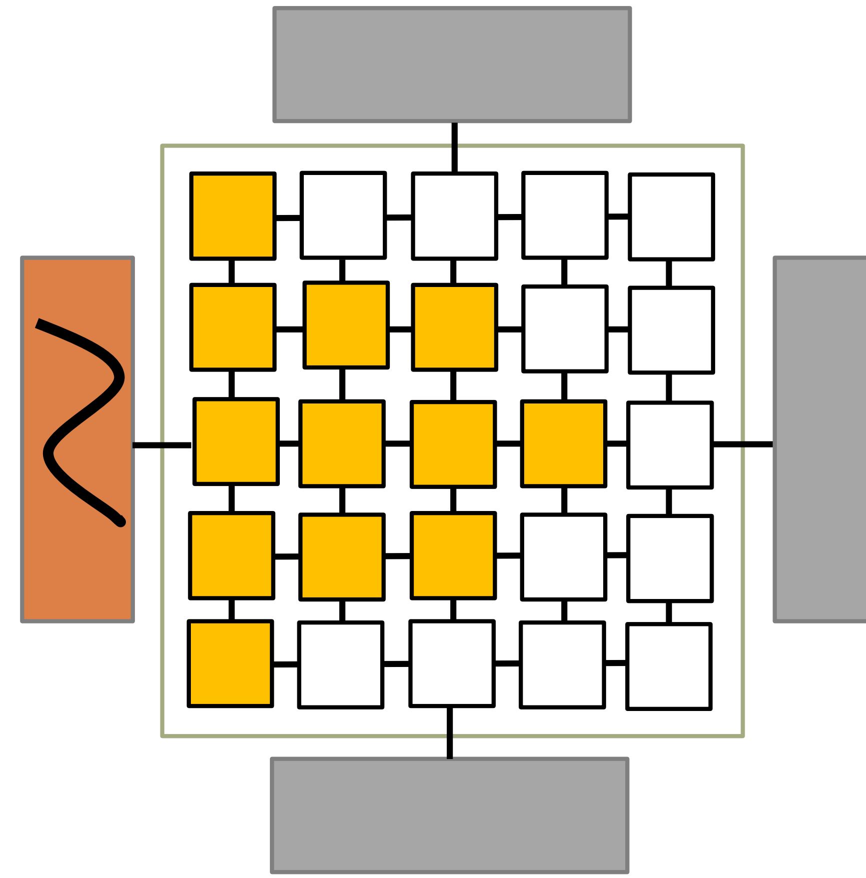
✓ Simple mechanisms

Dynamic policies can reduce data movement

App: Delaunay triangulation



Static NUCA



Jigsaw

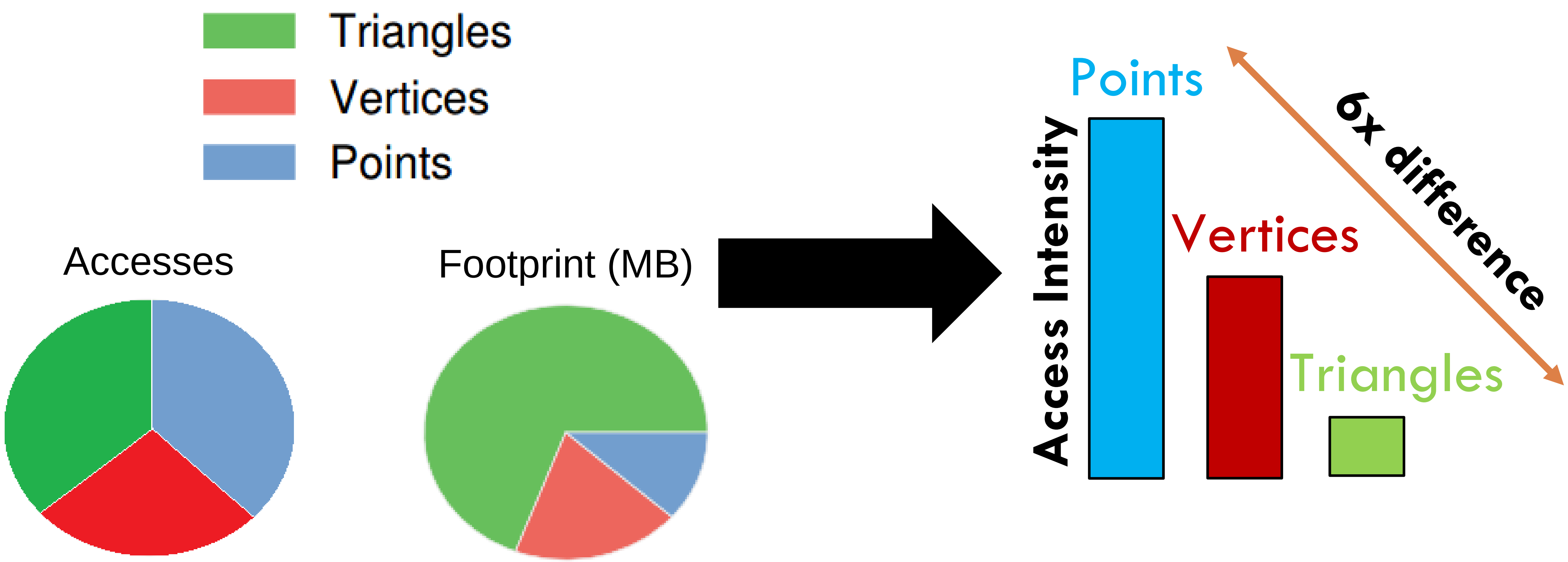
[Beckmann, PACT'13]

Dynamic policy performs somewhat better:

4% better performance

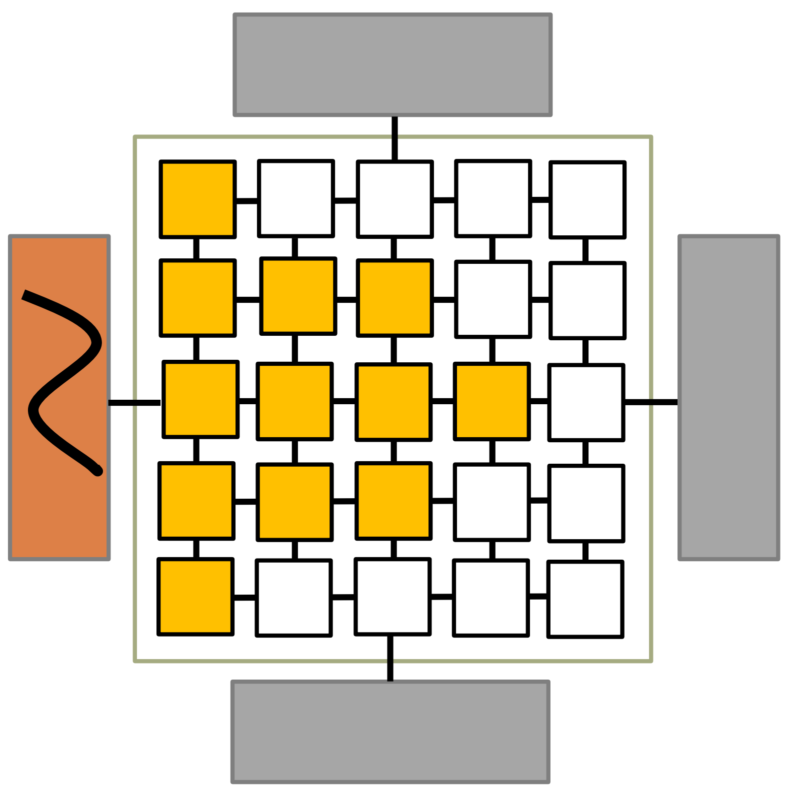
12% lower energy

Static analysis can help!



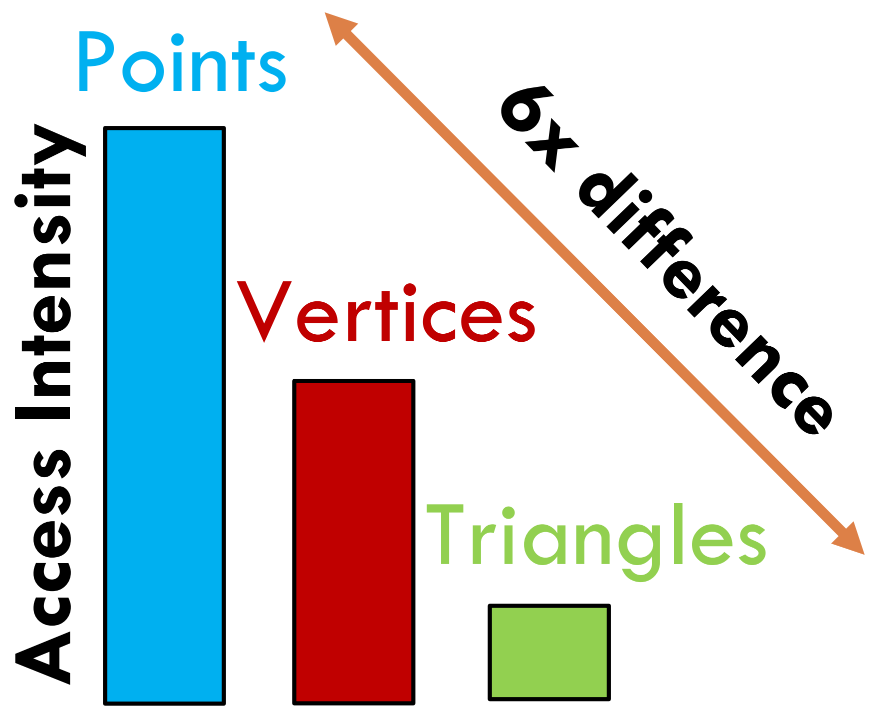
Jigsaw with Static Classification

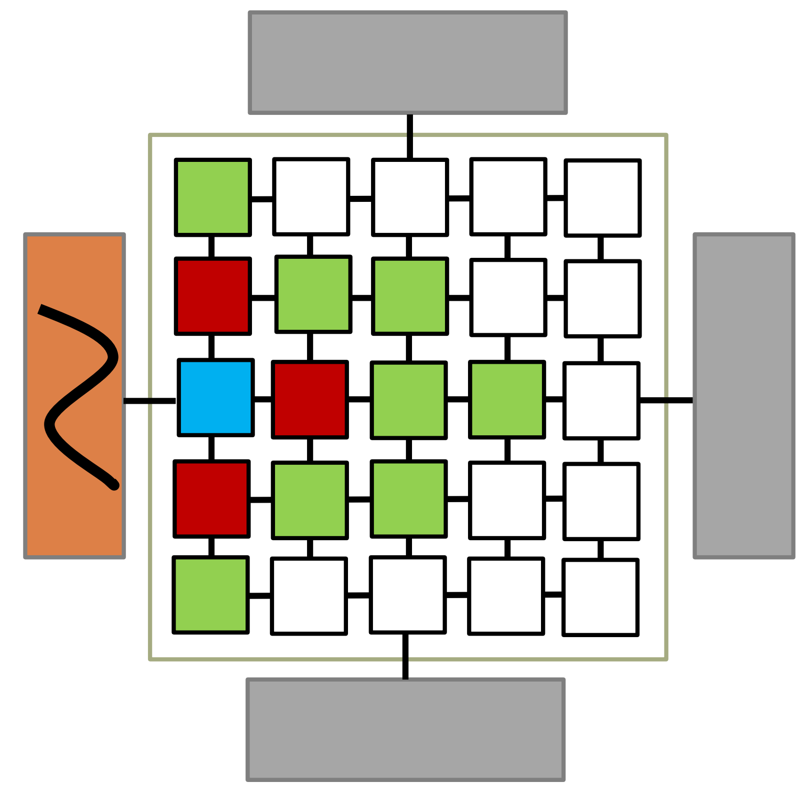
Few data structures accessed
more frequently than others



Jigsaw

[Beckmann, PACT'13]





Whirlpool!

Vs Jigsaw:

19% better performance

42% lower energy

Agenda

- Case study
- Manual classification
- Parallel applications
- WhirlTool

Whirlpool – Manual classification

Organize application data into memory pools

Points, Triangles

```
int poolPoints = pool_create();
Point* points = pool_malloc(sizeof(Point)*n, poolPoints);

int poolTris = pool_create();
Tri* smallTris = pool_malloc(sizeof(Tri)*m, poolTris);

Tri* largeTris = pool_malloc(sizeof(Tri)*M, poolTris);
```

Insight: Group semantically similar data into a pool

Minor changes to programs

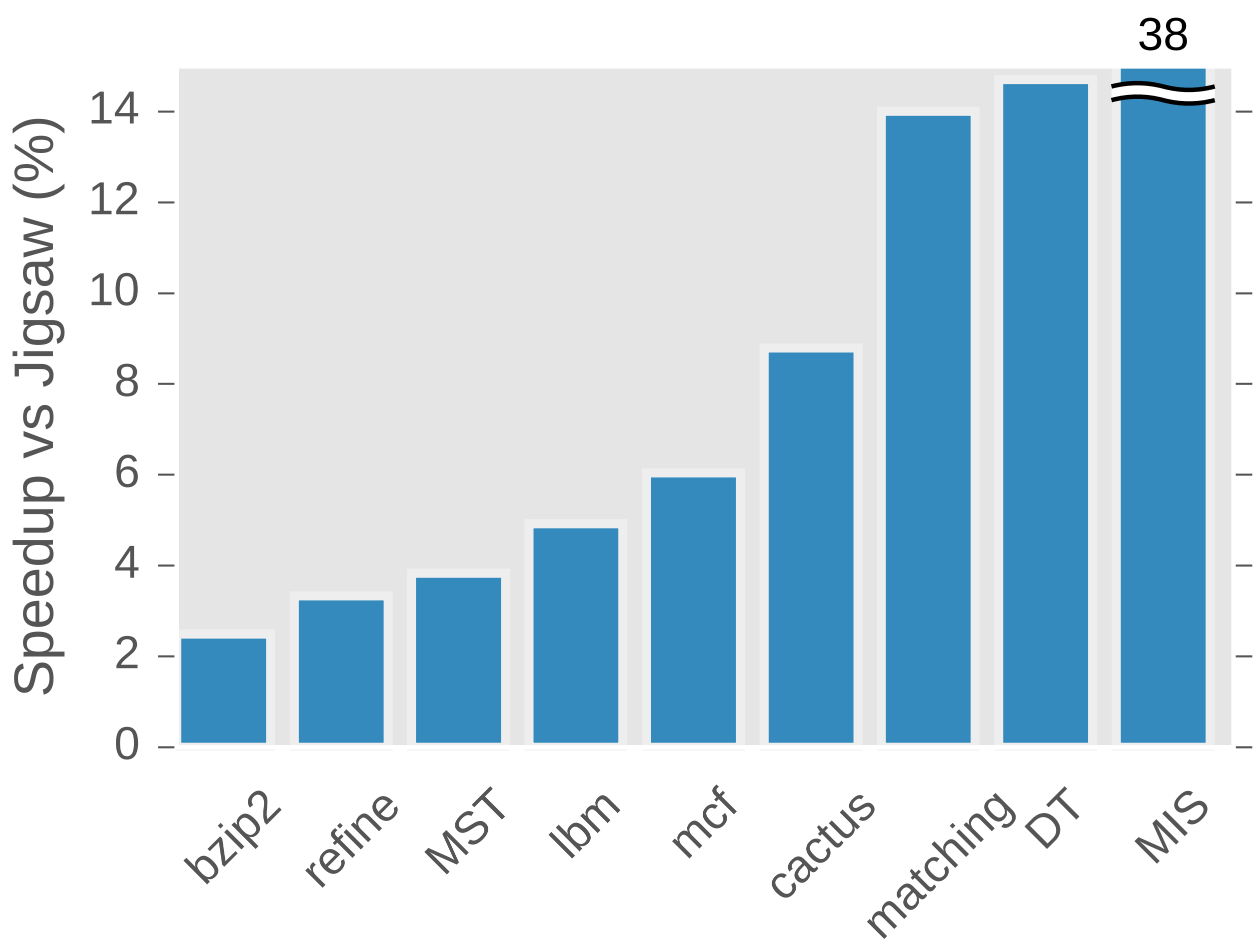
	Application
PBBS	Delaunay triangulation
	Maximal matching
	Delaunay refinement
	Maximal independent set
	Minimal spanning forest
SPECCPU 2006	401.bzip2
	470.lbm
	429.mcf
	436.cactusADM

Whirlpool on NUCA placement

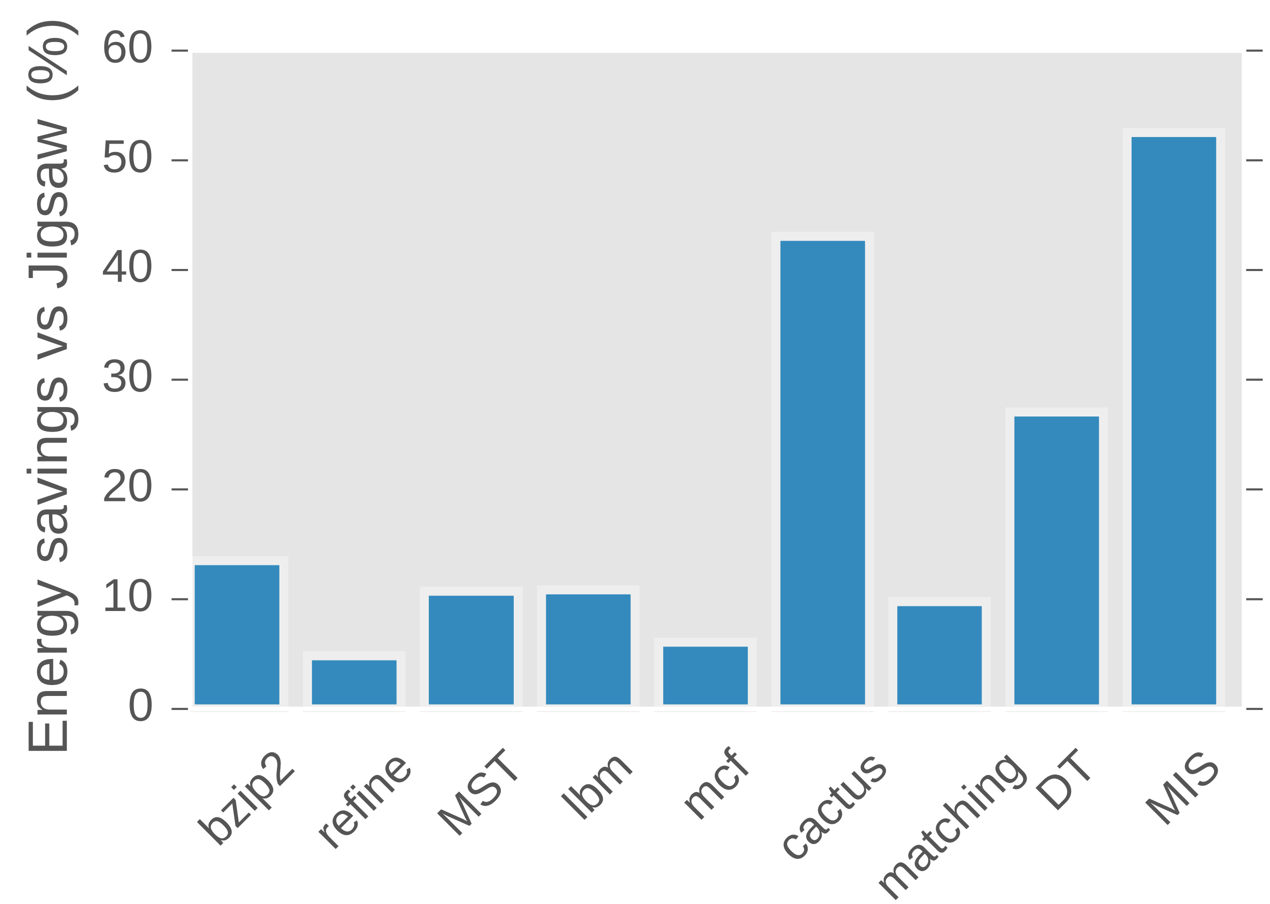
- Use pools to improve Jigsaw's decisions
  - Each pool is allocated to a virtual cache
  - Jigsaw transparently places pools in NUCA banks
- Whirlpool requires no changes to core Jigsaw
  - Increase size of structures (few KBs)
  - Minor improvements, e.g. bypassing (see paper)
- Pools useful elsewhere, eg to dynamic prefetching

Significant improvements on some apps

Performance



Energy



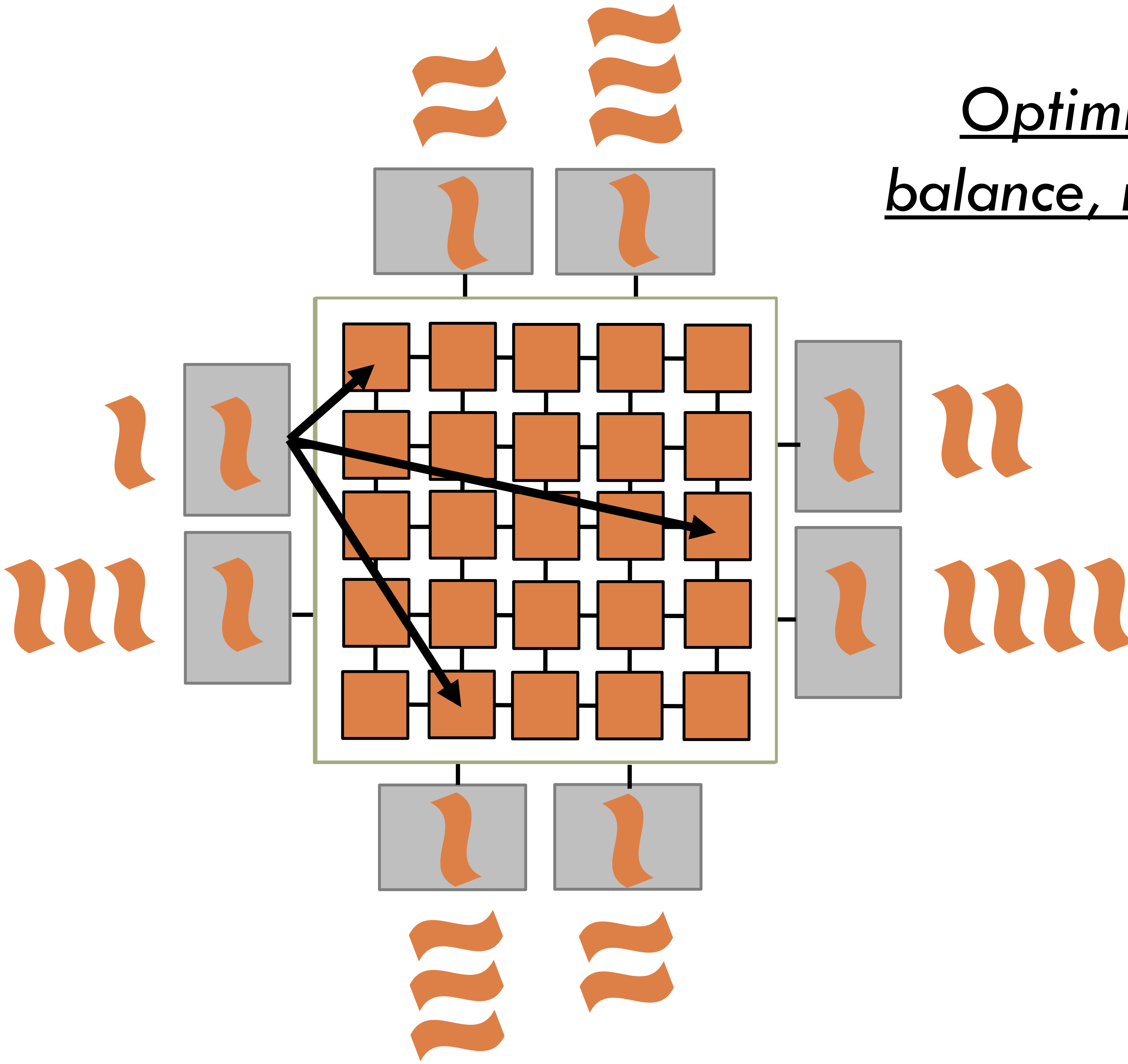
Up to 38% better performance

Up to 53% lower energy

Agenda

- Case study
- Manual classification
- Parallel applications
- WhirlTool

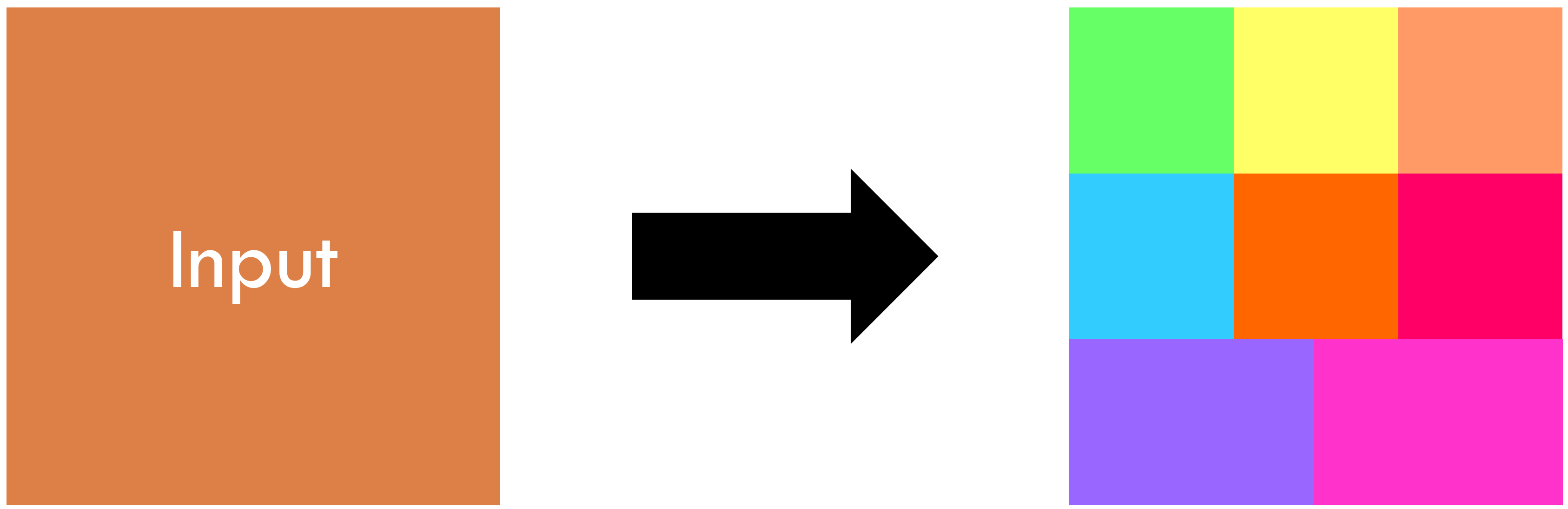
Conventional runtimes can harm locality



Optimize load balance, not locality

Whirlpool co-locates tasks and data

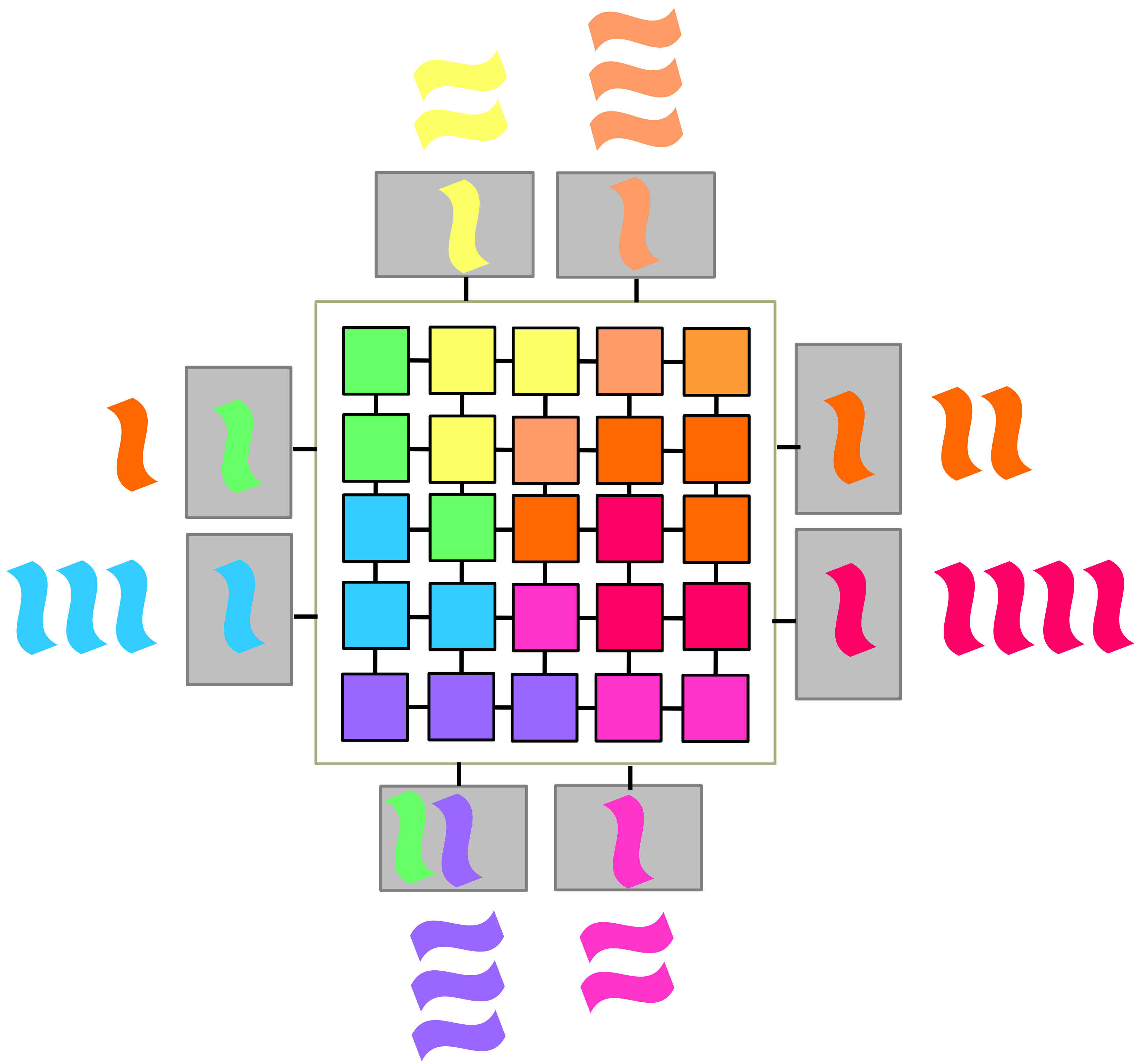
- Break input into ***pools***



- Application indicates task affinity
- Schedule + steal tasks from nearby their data
- Dynamically adapt data placement

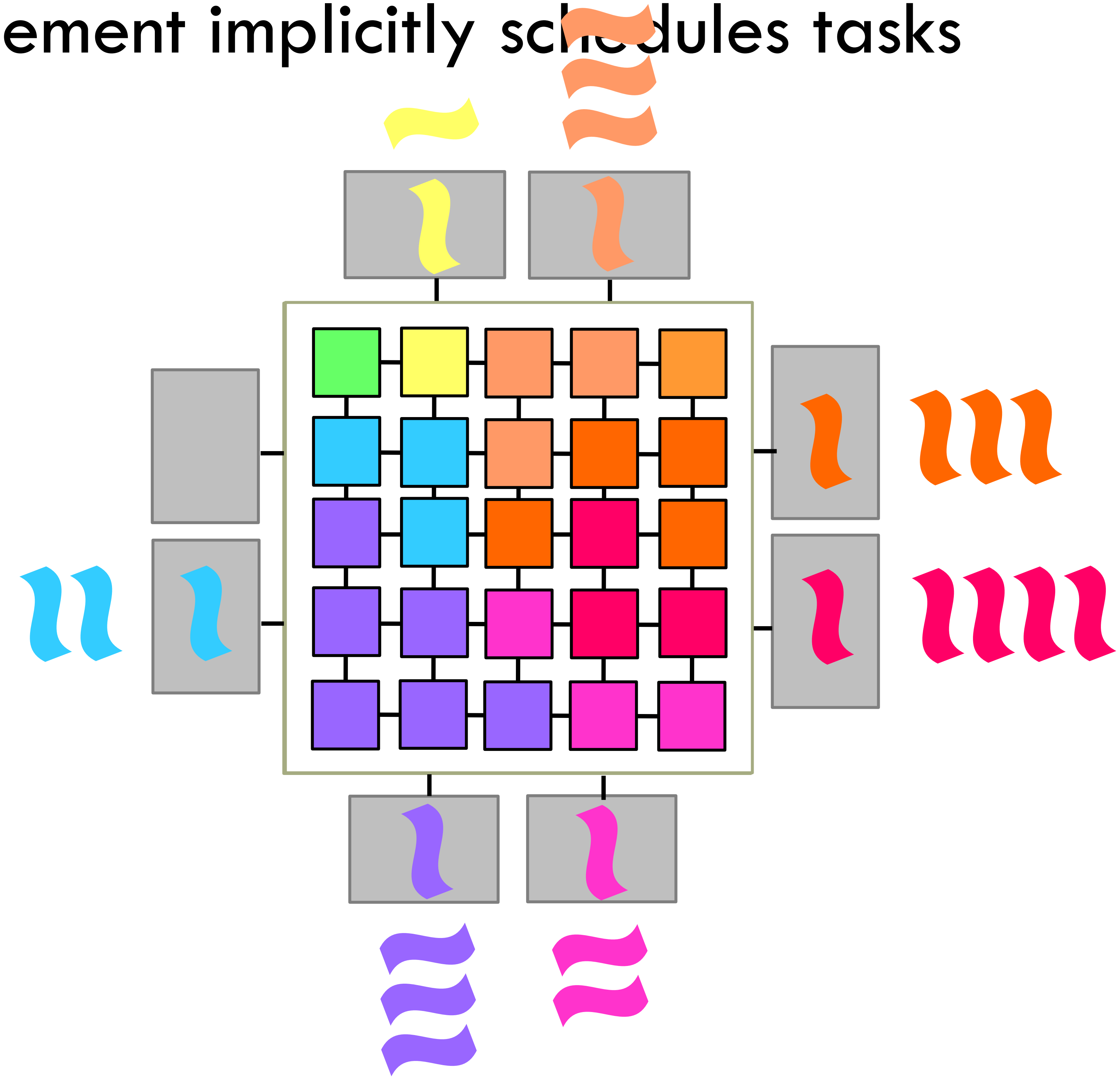
- Requires minimal changes to task-parallel runtimes

Whirlpool improves locality



Whirlpool adapts schedule dynamically

- Data placement implicitly schedules tasks



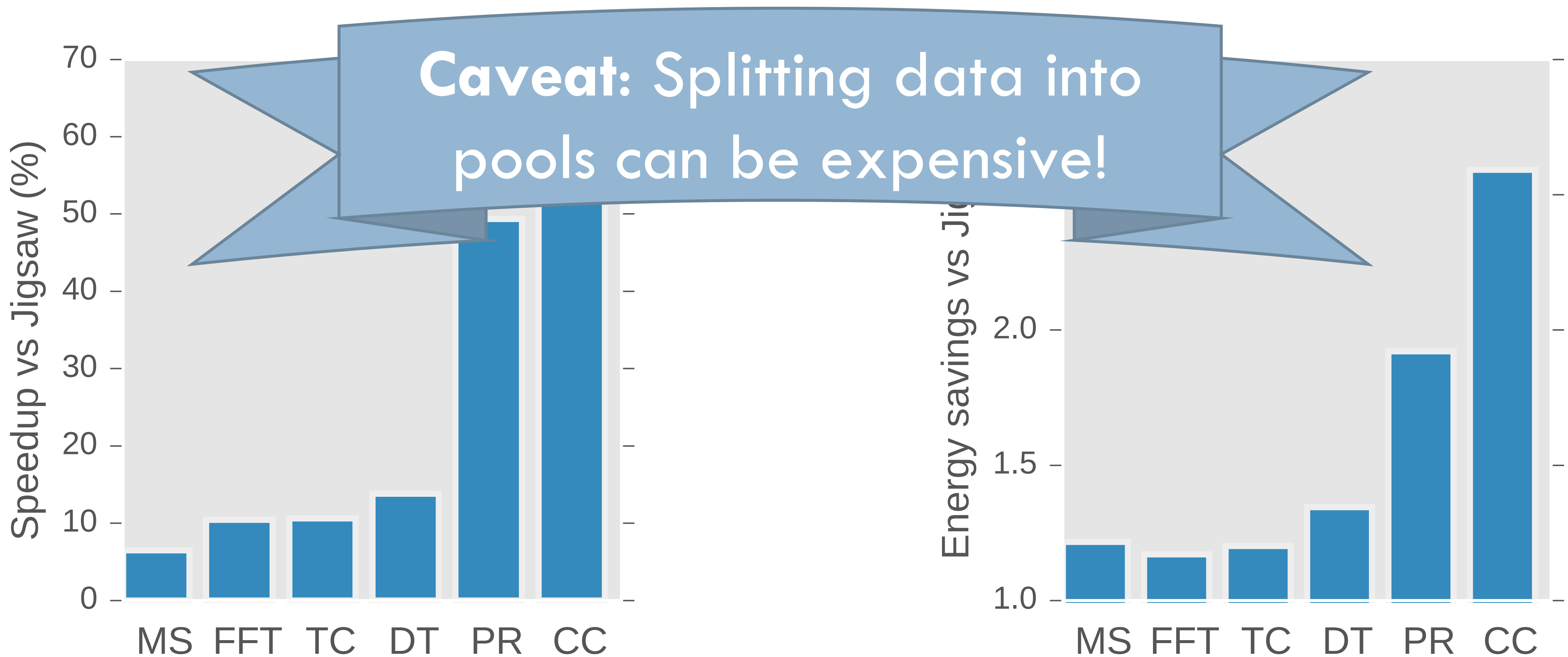
Significant improvements at 16 cores

Applications

Divide and conquer algorithms: Mergesort, FFT

Graph analytics: PageRank, Triangle Counting, Connected Components

Graphics: Delaunay Triangulation



Up to 67% better performance

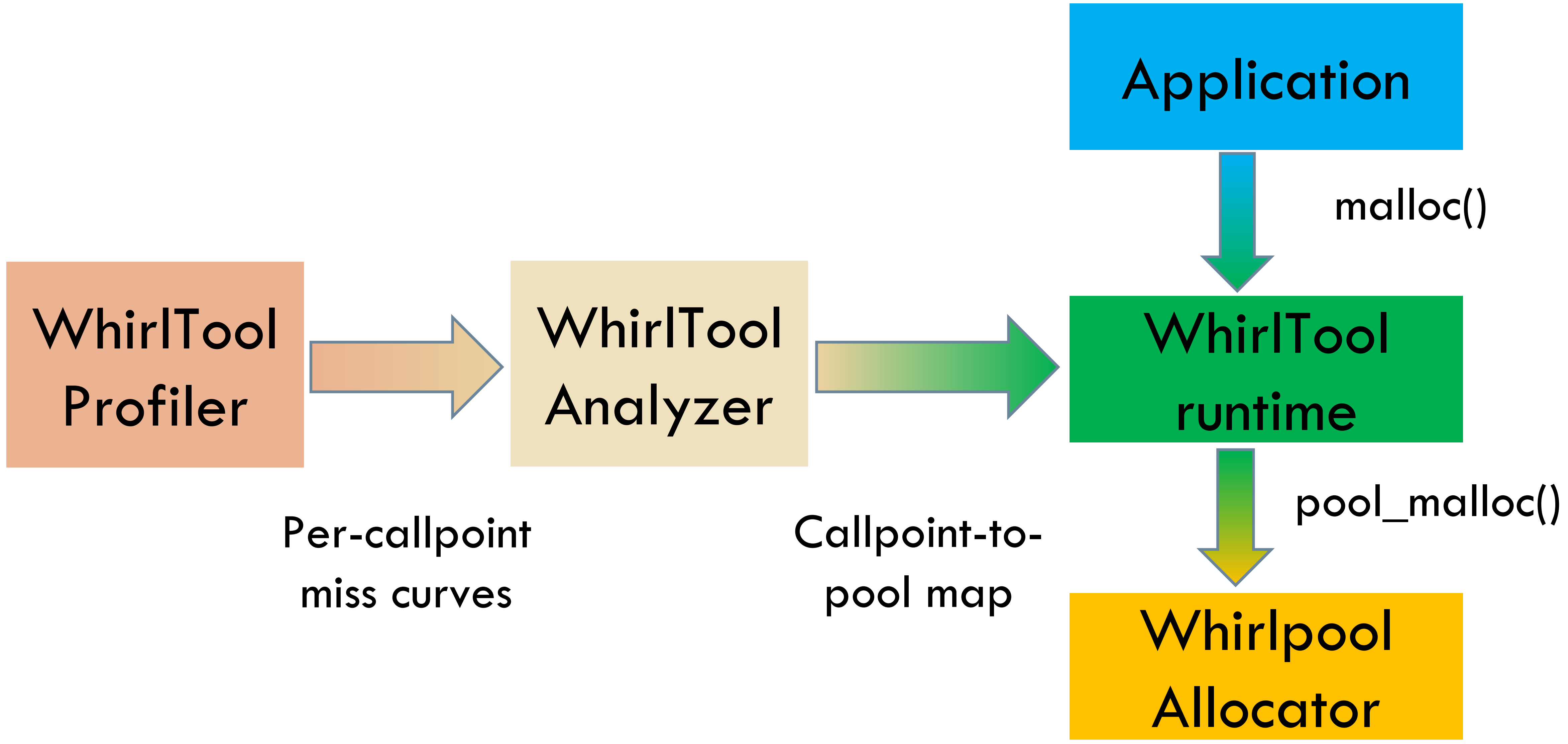
Up to 2.6x lower energy

Agenda

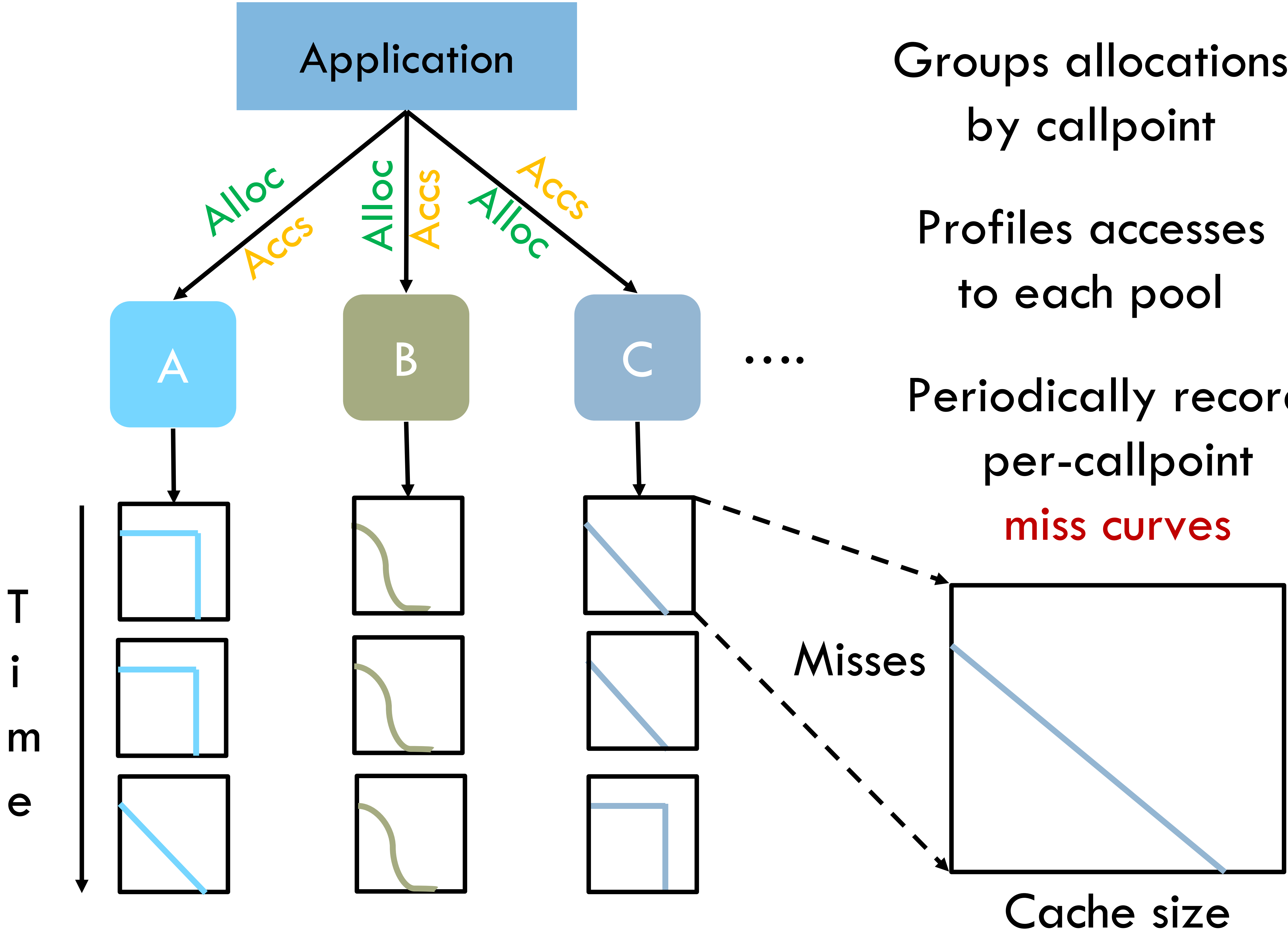
- Case study
- Manual classification
- Parallel applications
- WhirlTool

WhirlTool – Automated classification

- Modifying program code is not always practical
- A profile-guided tool can automatically classify data into pools



WhirlTool profiles miss curves



Groups allocations by callpoint

Profiles accesses to each pool

Periodically records per-callpoint miss curves

WhirlTool analyzes curves to find pools

- Hardware can only support a limited number of pools
  - Jigsaw uses 3 virtual caches / thread
    ➔ 0.6% area overhead over LLC
  - Whirlpool adds 4 pools (each mapped to a virtual cache)
    ➔ 1.2% total area overhead over LLC

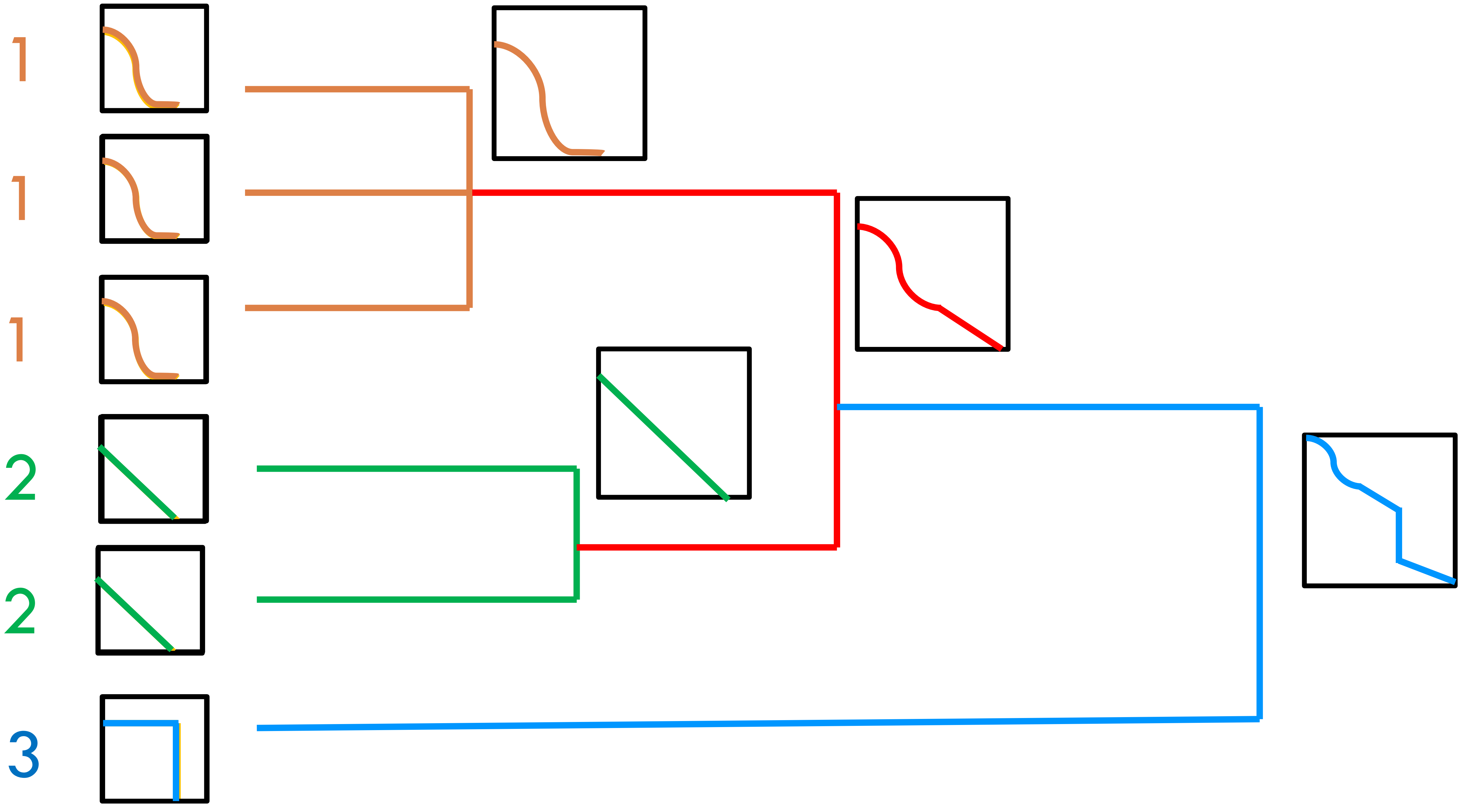
- Must cluster callpoints into semantically similar groups

Per-callpoint miss curves

Agglomerative clustering

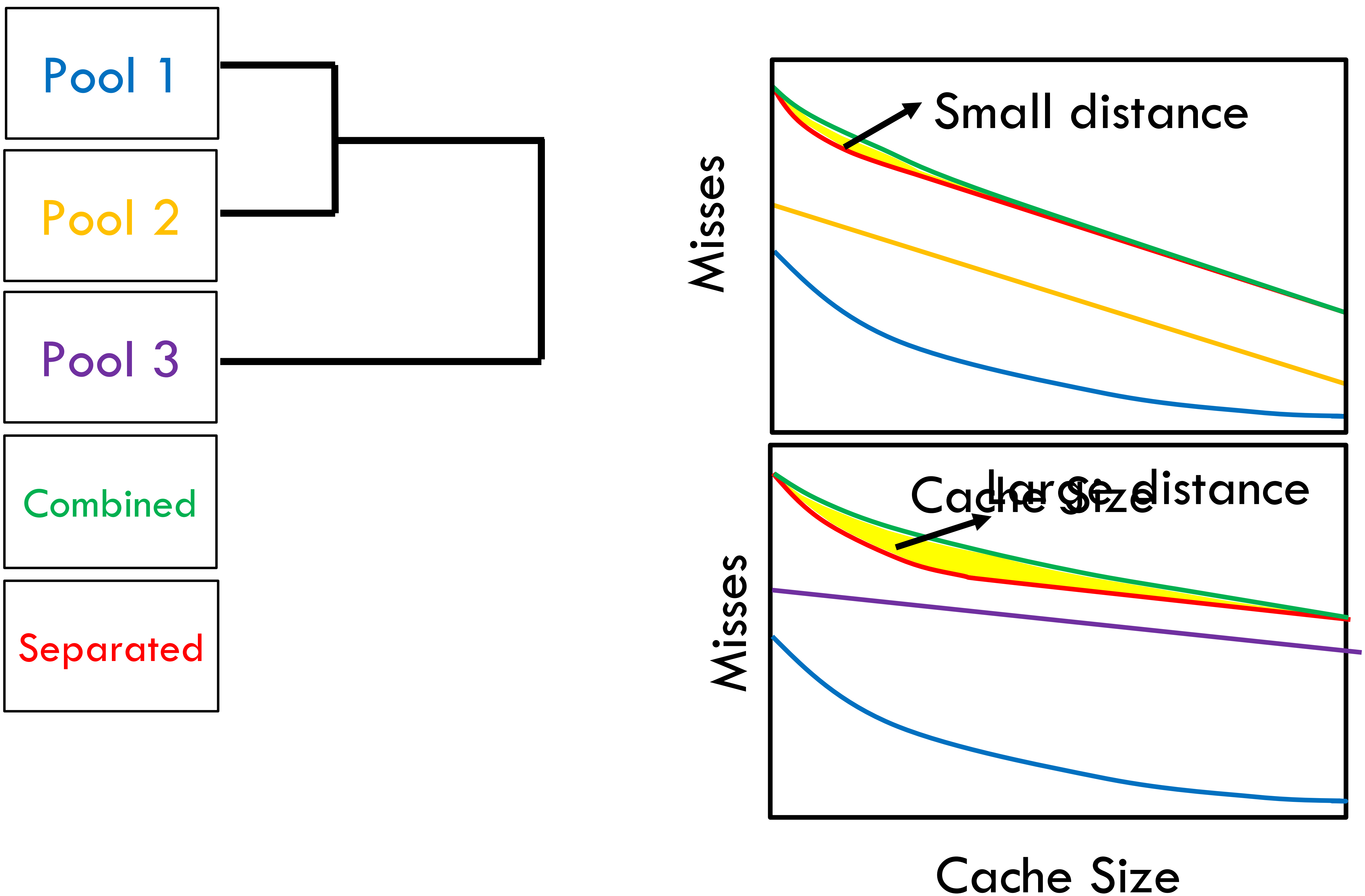
Callpoint-to-pool mapping

Example of agglomerative clustering

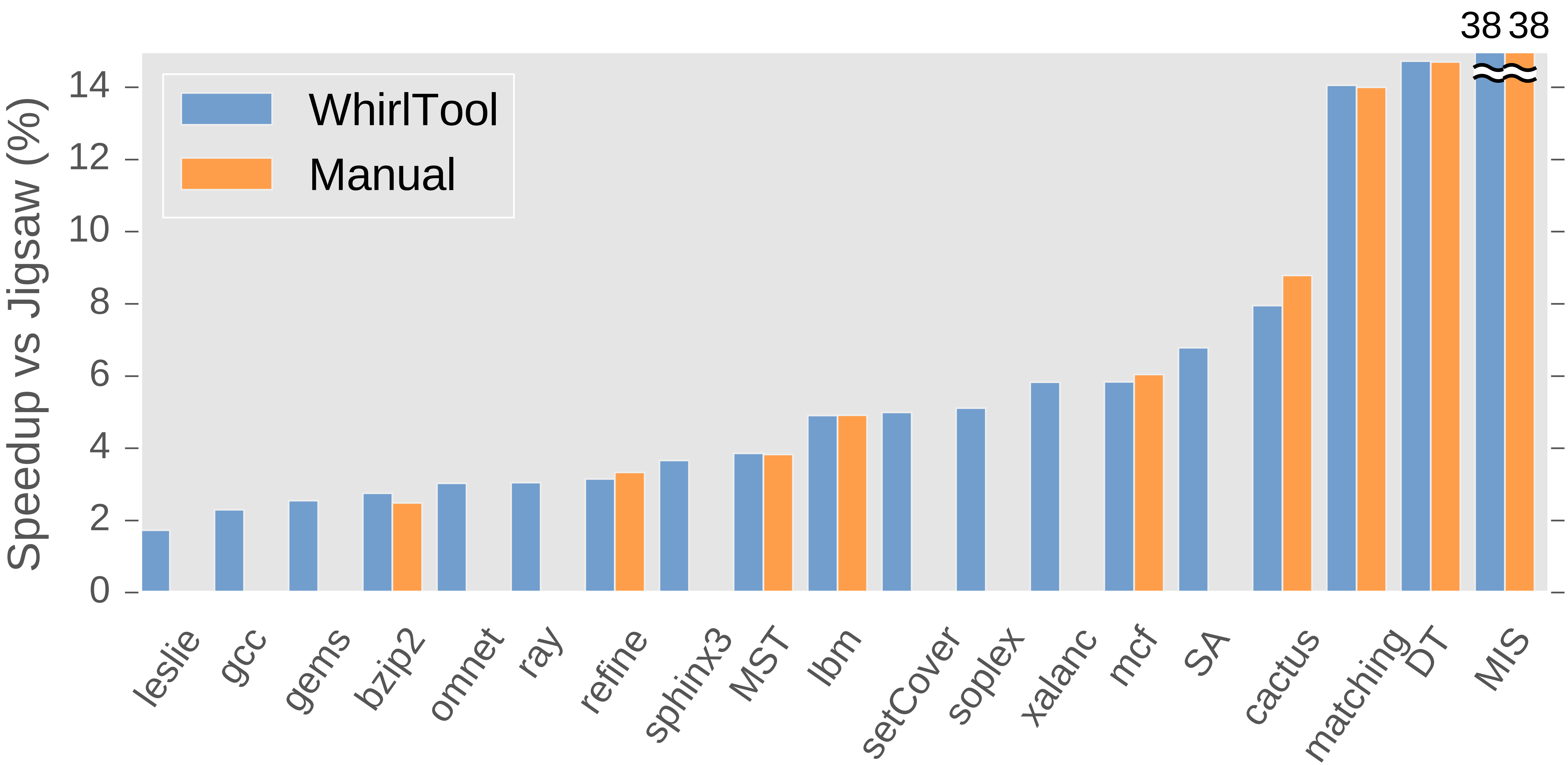


WhirlTool's distance metric

How many misses are saved by separating pools?



WhirlTool matches manual hints



Multiprogram mixes

- 4-core system with random SPECCPU2006 apps
  - Including those that do not benefit

- Whirlpool improves performance by (gmean over 20 mixes)
  - 35% over S-NUCA
  - 30% over idealized shared-private D-NUCA *[Hererro, ISCA'10]*
  - 26% over R-NUCA *[Hardavellas, ISCA'09]*
  - 18% over page placement by Awasthi et al. *[Awasthi HPCA'09]*
  - 5% over Jigsaw *[Beckmann, PACT'13]*

Conclusion

- Semantic information from applications improves performance of dynamic policies
- Coordinated data and task placement gives large improvements in parallel applications
- Automated classification reduces programmer burden

Thanks For Your Attention!

Questions are welcome!

WhirlTool code available at http://bit.ly/WhirlTool

Massachusetts Institute of Technology

CSAIL