Multiple Page Size Modeling and Optimization

Calin Cascaval, Evelyn Duesterwald, Peter F. Sweeney, and Robert W. Wisniewski
IBM TJ Watson Research Center

Multiple Page Size Support

Benefits and costs:

- Large pages reduce page fault and TLB misses
- Large pages improve hardware prefetch
- Mapping everything to large pages may hurt performance (increased swapping overhead)

Current situation:

- OS partitions memory into pools of small and large pages
- OS offers ability to back an application's heap and static data with large pages
- Requires programmer effort to select applications that benefit from large pages and modify the application and environment accordingly
- Limited OS work to upgrade pages with heuristics (not based on modeling benefits)

CPO page size agent benefits:

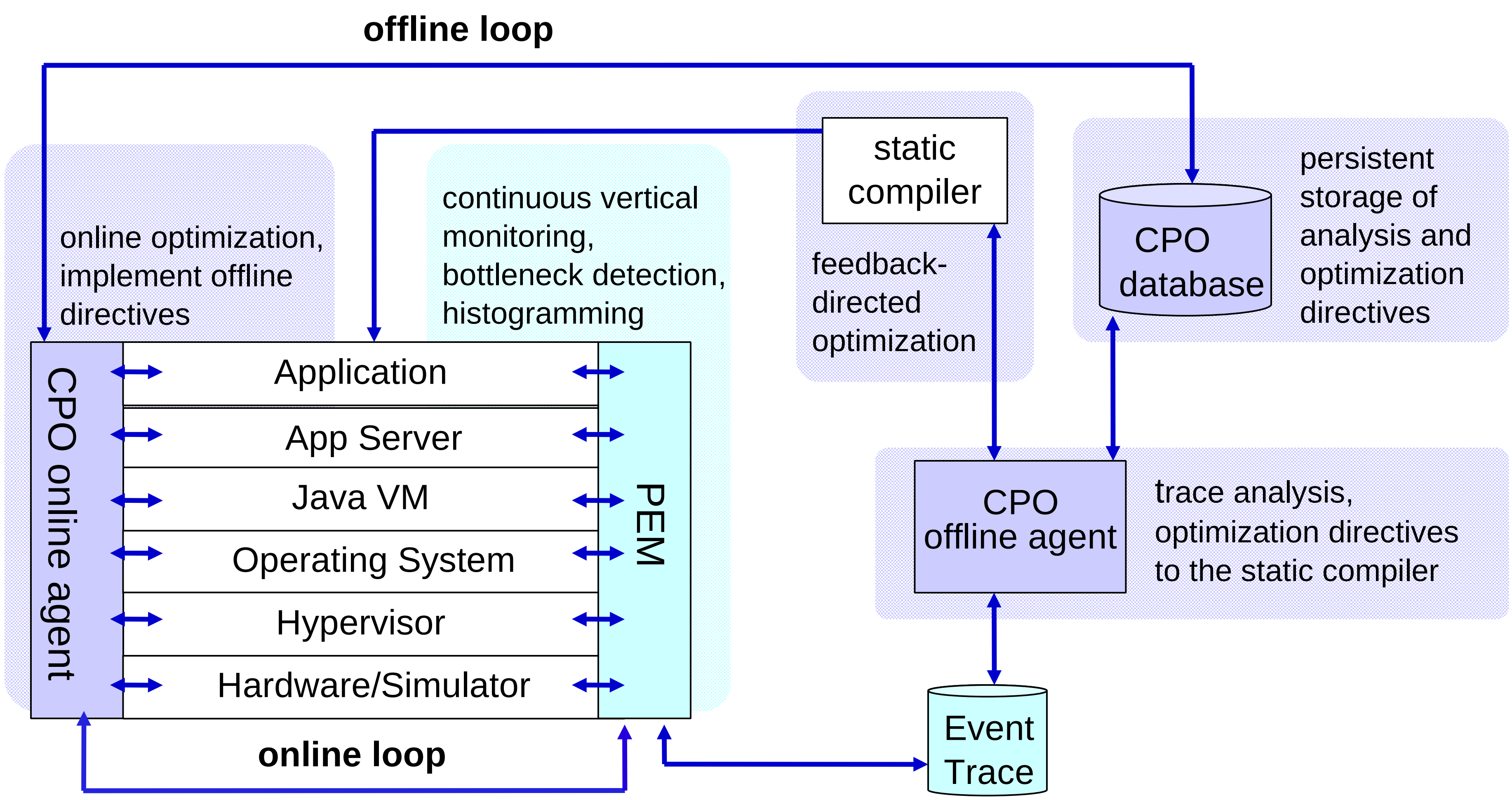
- Automatically achieve improvements from large pages without programmer intervention
- Explore the benefits of finer (per data structure) data space partitioning for large page mapping
- Allows data structures layout to be re-arranged

Outline

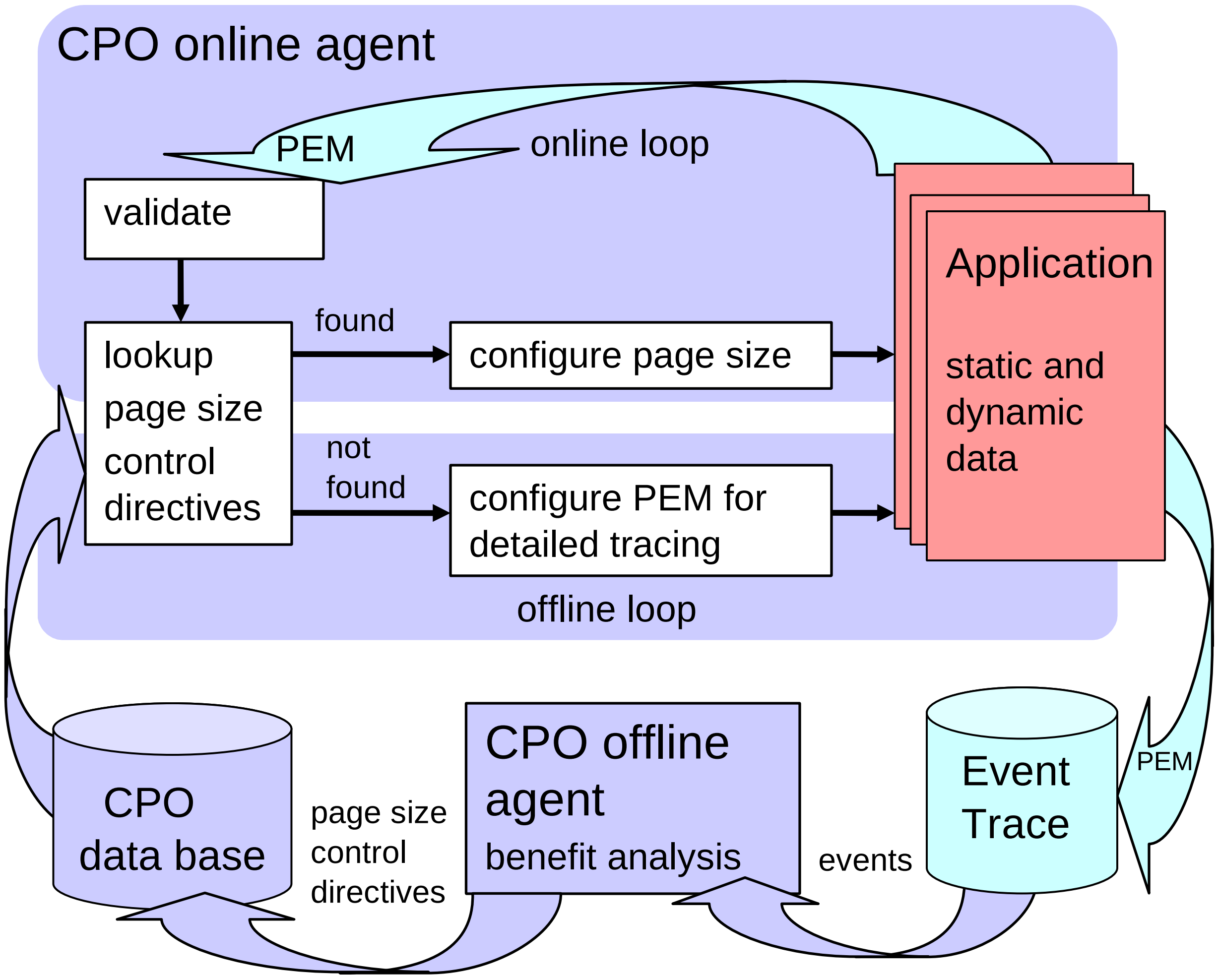
- **CPO vision**
- **CPO page size agent:**
  - Multiple page size benefit analysis
  - Experimental results
- **Conclusions**

Continuous Program Optimization Vision

Continuous offline and online optimization by tuning across execution layers

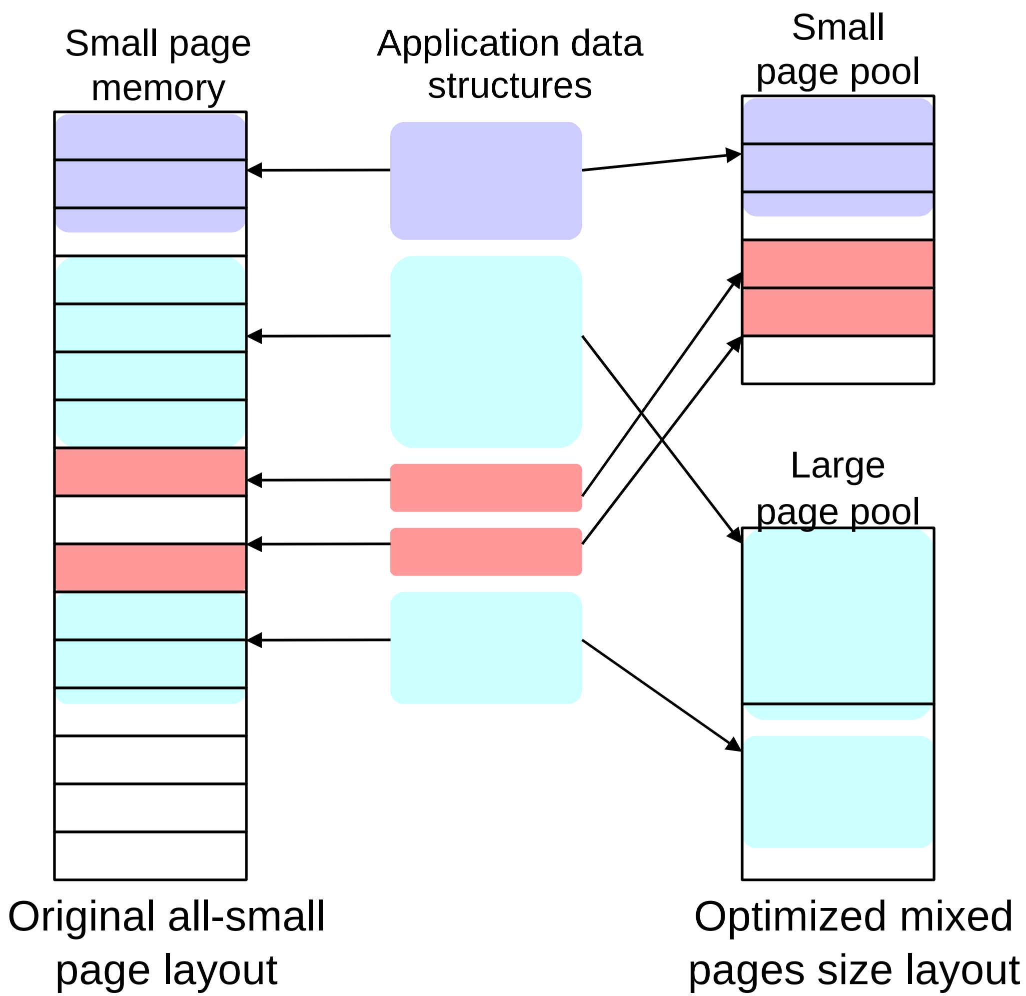


CPO Page Size Agent



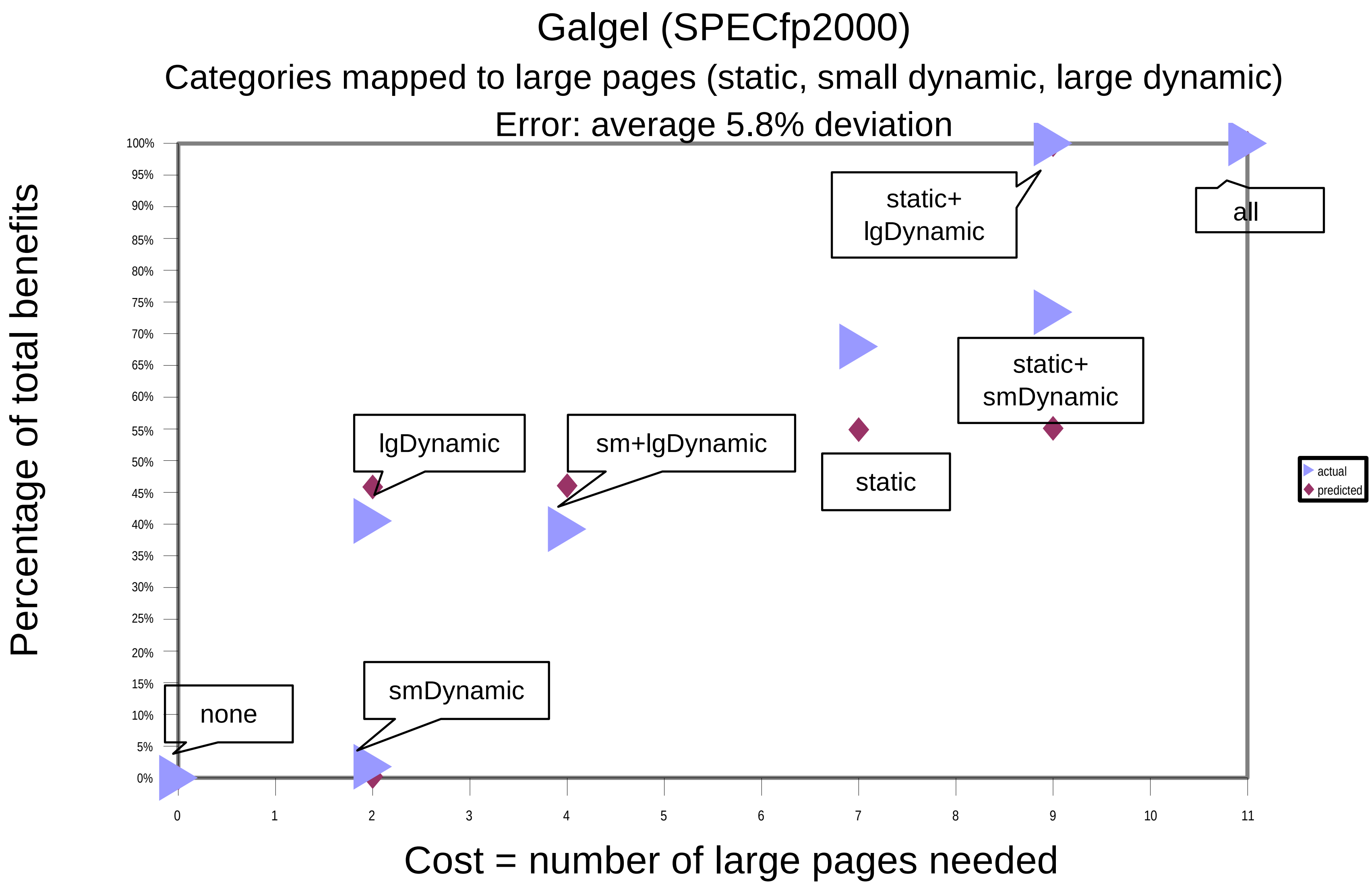
- Fully automated
- Offline CPO agent: select profitable data structures for large pages
- Online CPO agent: map selected data structures to large pages

Large Page Benefit Analysis -- Problem Definition



- **Partition data structures into categories, e.g.:**
  - Large dynamic data structures
  - Small dynamic data structures
  - Static data
- **Explore the most promising mappings of categories to page size**
- **Calculate a cost/benefit ranking:**
  - Number of large pages needed
  - Reduction in page faults and TLB miss cycles
- **Select mapping with highest benefits for available large pages**

Large Page Benefit Analysis Evaluation

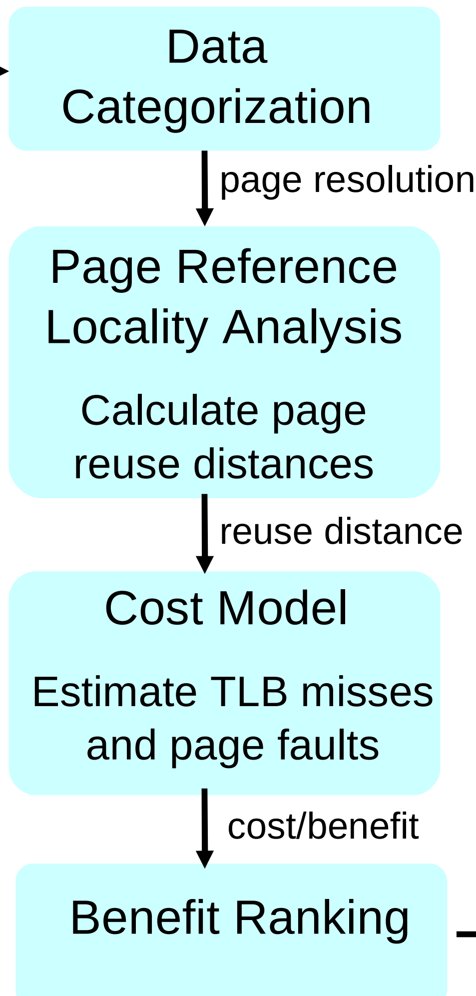


Large Page Benefit Analysis -- Algorithm overview

Input:

PEM trace with

- Sampled load store addresses from HW counters
- Data structure allocation events



Set of ranked large page mappings

The calculated benefit ranking approximates actual benefits

- **Based on sampled data reference addresses**
- **Cost model does not simulate memory hierarchy and prefetch**

Ranking is a relative measure

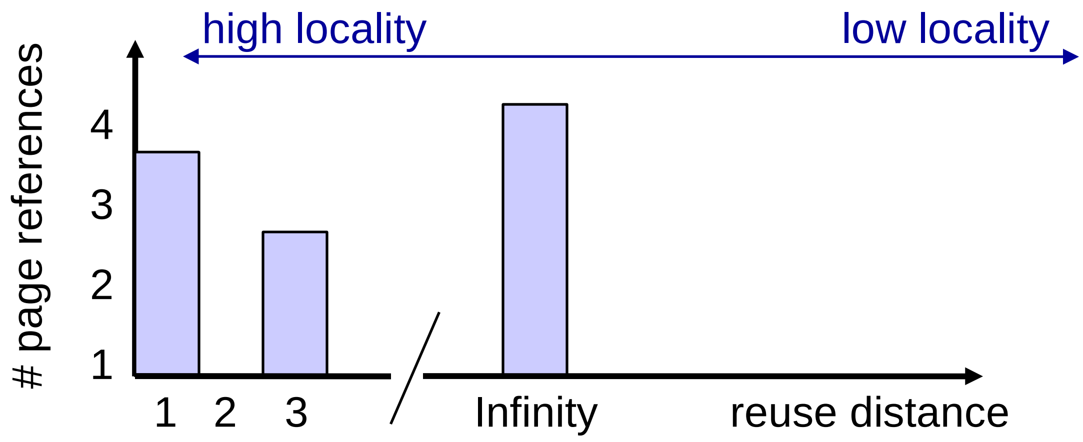
Background: Page Reuse Distance

LRU page reuse distance *RD* for a page reference stream: p_1, p_2, $p_{3, \ldots}$

- $RD(p_t)$ = #intervening unique page references since *p* was last referenced

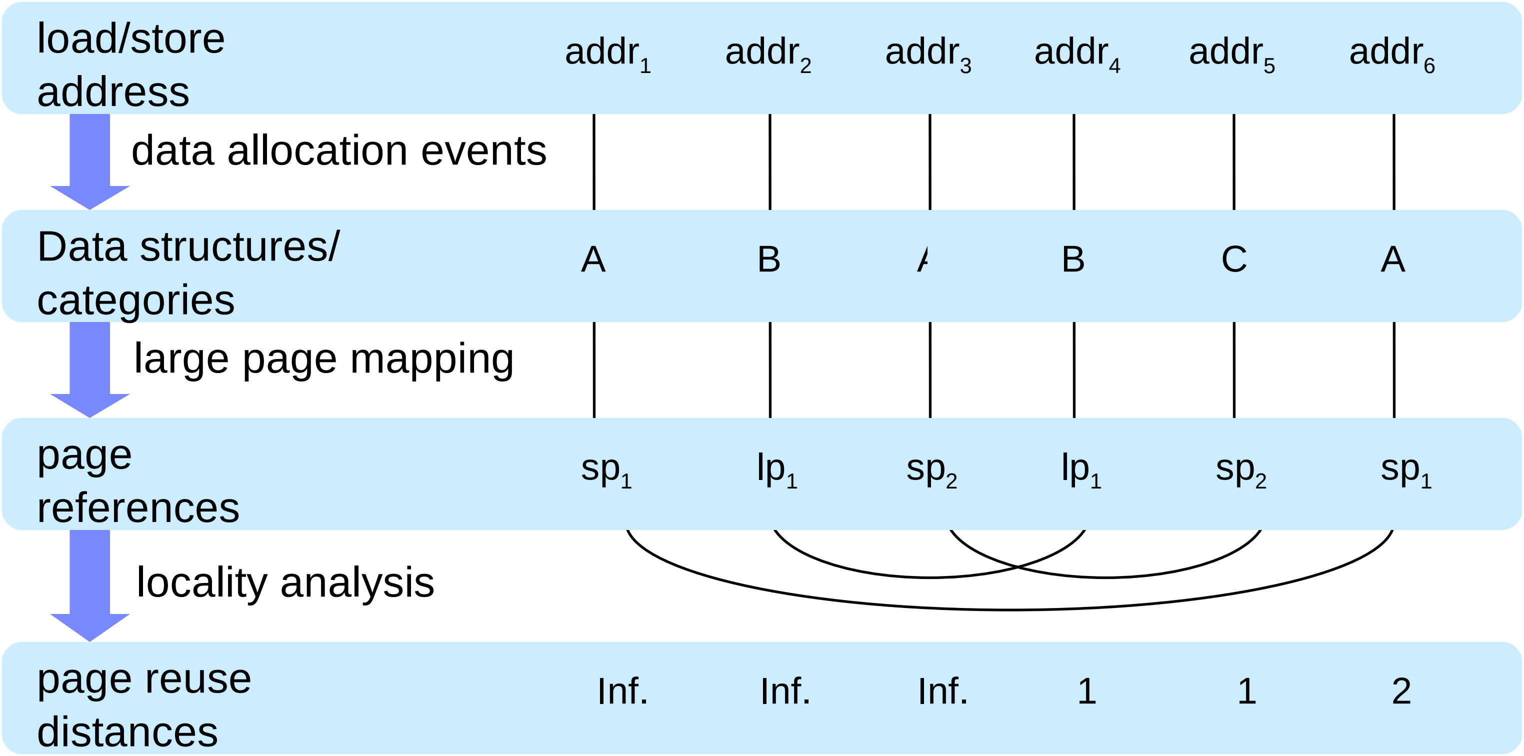
Page references:	A	B	C	B	D	B	A	C	A
Reuse distance:	Inf.	Inf.	Inf.	1	Inf.	1	3	3	1

LRU page reuse distance histogram



Page Reference Locality Analysis

- **LRU page reuse distance *RD* for a page reference stream: p_1, p_2, $p_{3, \ldots}$**
- **$RD(p_t)$ = #intervening unique page references since *p* was last referenced**



Large Page Cost Model

Benefit of a mapping *m* is the reduction in cycles over the "*allSmall*" page mapping:

$$Benefit_m = (MissCycles_{allSmall} - MissCycles_m) * SamplingScalingFactor$$

where for any mapping *i:*

$$MissCycles_i = TLBmiss_i * Cost(TLBmiss) + Pgflts_i * Cost(pgflt)$$

$$TLBmiss_i = | \{ \text{r} \quad Ref_{\text{TLB}} \mid RD_{TLB}(r) \geq \text{ number of TLB entries} \} |$$

$$Pgflts_i = | \{ \text{r} \quad Ref_{\text{PT}} \mid RD_{PT}(r) \geq \text{ number of page table entries} \} |$$

Ref_{TLB} = TLB references

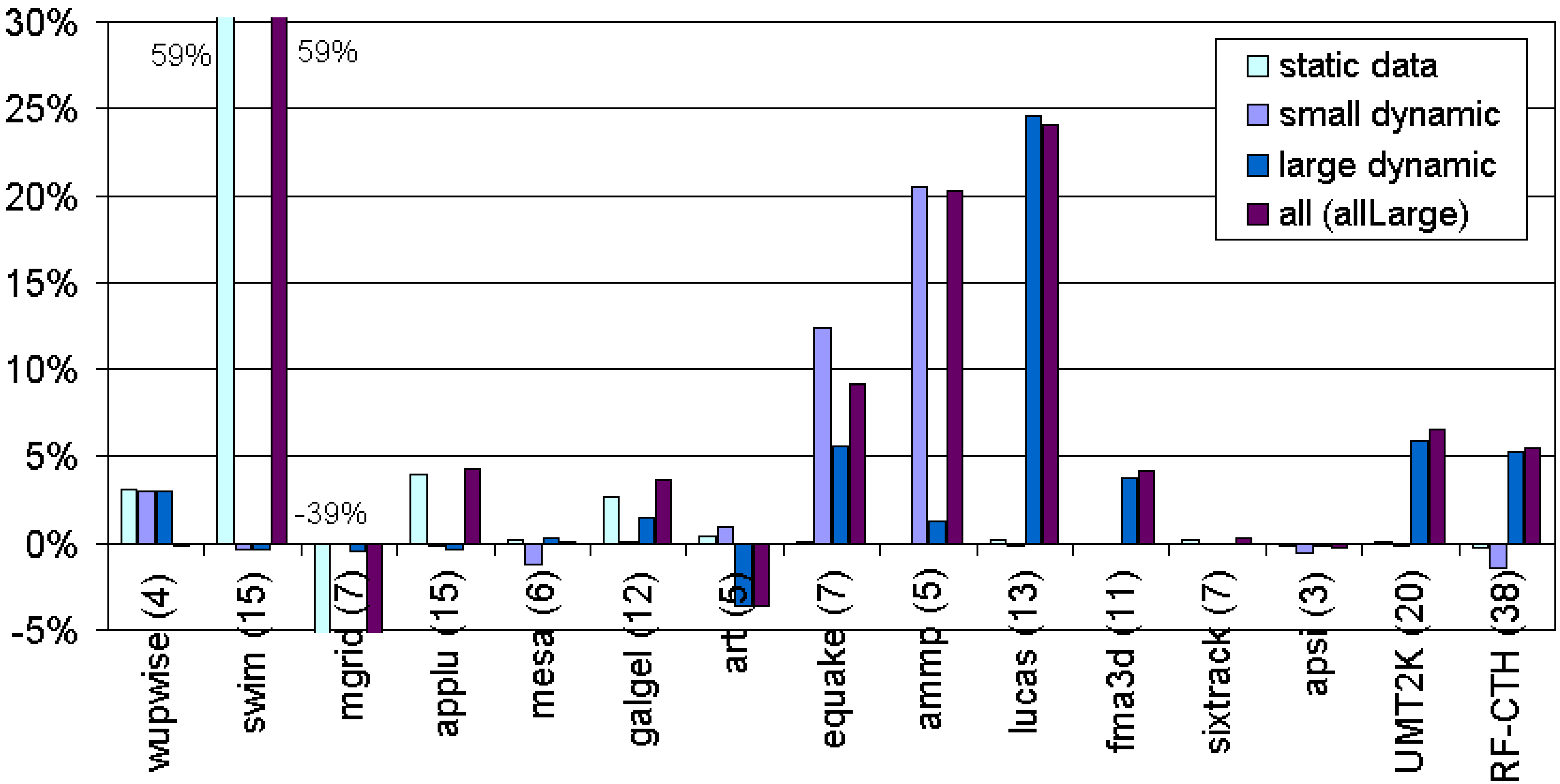
Ref_{PT} = Page Table references = TLB reference that result in a TLB miss

Experimental Methodology

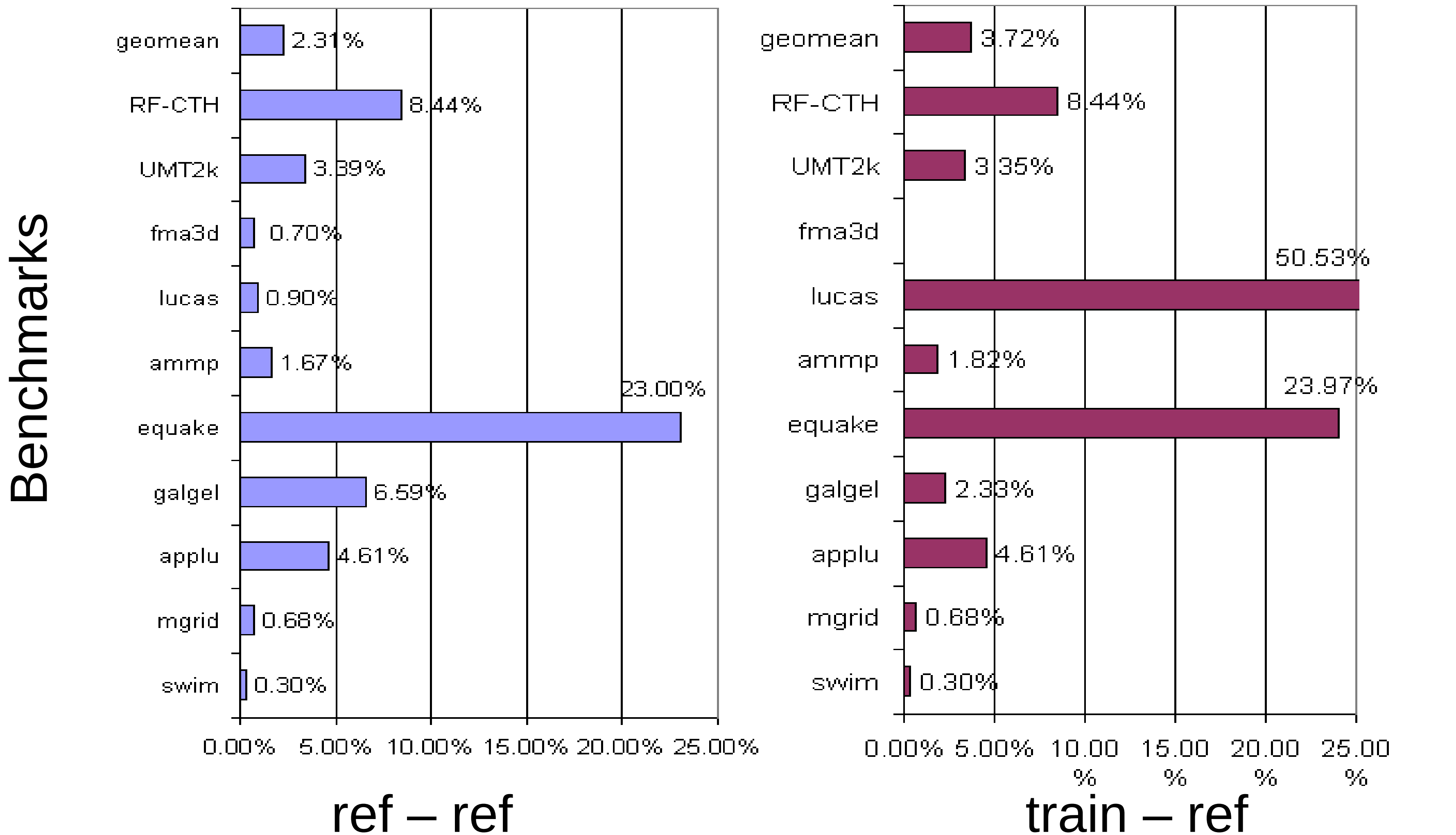
- K42 open-source operating system
- PowerPC 970FX Apple Xserve
- SPECfp2000 and 2 large scientific codes, UMT2K and RF-CTH
- Median of 5 runs with all mappings
- Filter out benchmarks that do not have a significant number of TLB misses and page faults

Large Page Benefits – Speedups

SPECfp2000 + UMT2K + RF-CTH



Large Page Benefit Analysis – PMB Error Margins
(PMB = Percentage of maximum benefit)



Related Work

- **OS large page migration**
  - Taluri94, Romer95, Navarro02
- **Reuse distance for modeling memory behavior**
  - Mattson70, etc.
- **Continuous Program Optimization**
  - Dynamic optimization
    - Arnold05
  - Continuous compilation
    - Plezbert97, Kistler03, Childers03.

Conclusions

- **We presented a model for estimating the performance benefits for mapping data categories to different page sizes**
- **We have shown how the model can be integrated in the CPO framework as an online agent to drive page mapping decisions at run-time**
- **We validated the model on a real system using data collected through PEM from the entire execution stack**