BORG: Block-reORGanization for Self-optimizing Storage Systems

Medha Bhadkamkar Jorge Guerra Luis Useche
Sam Burnett Jason Liptak
Raju Rangaswami Vagelis Hristidis

Florida International University

March 9, 2009

Problem

- I/O is the bottleneck
  - ✓ Legacy filesystems favor sequential access.
  - ✓ Realistic workloads are not necessarily sequential
- Proposed Solution
  - ✓ Co-locate data based on workload block access patterns
  - ✓ Improve sequentiality

Workload Characteristics that motivate BORG

- Workloads
  - ✓ office - browser, OpenOffice applications, gnuplot, etc
  - ✓ developer - emacs, gcc, gdb, etc
  - ✓ Subversion (SVN) server - Sources and document repository
  - ✓ Web server - Department web server
- Workloads Statistics Summary

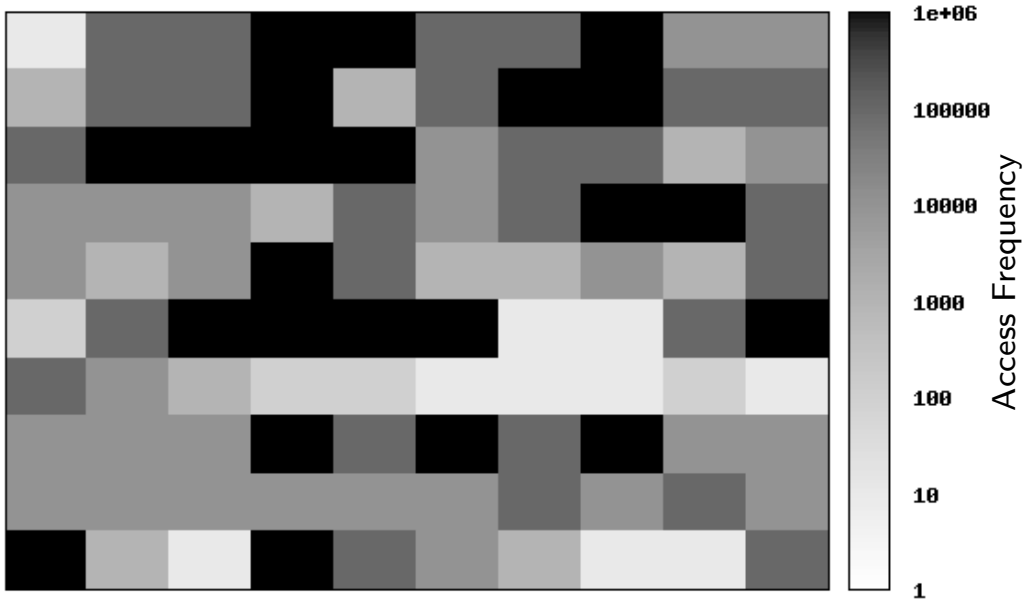
Workload type	**File System size [GB]**	**Total [GB] Reads**	**Total [GB] Writes**
office	8.29	6.49	0.32
developer	45.59	3.82	10.46
SVN server	2.39	0.29	0.62
web server	169.54	21.07	2.24

Non-uniform Access Frequency Distribution

- Frequently accessed data is usually a small portion of the entire data.
- Frequently accessed data is spread over entire disk area

Workload type	**File System size [GB]**	**Unique [GB] Reads**	**Unique [GB] Writes**	**Top 20% data access**
office	8.29	1.63	0.22	51.40 %
developer	45.59	2.57	3.96	60.27 %
SVN server	2.39	0.17	0.18	45.79 %
web server	169.54	7.32	0.33	59.50 %

Non-uniform Access Frequency Distribution



The Opportunity

Co-locating frequently accessed data can improve I/O performance.

Workload Characteristics - Partial Determinism

- Non-sequential accesses repeat in a block access sequence

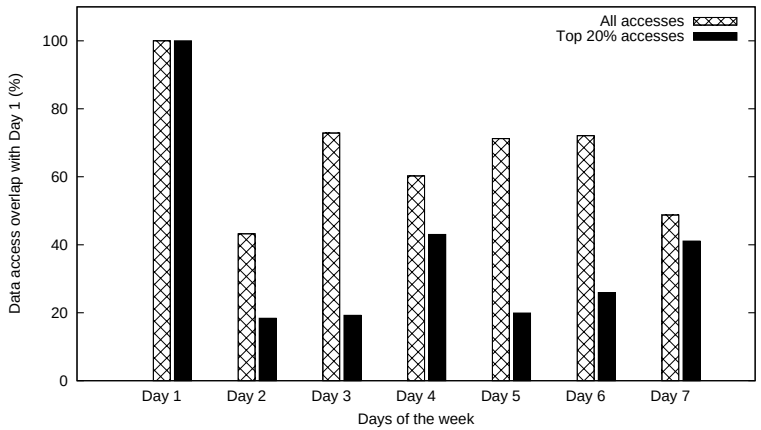
Workload type	**Partial determinism**
office	65.42 %
developer	61.56 %
SVN server	50.73 %
web server	15.55 %

The Opportunity

Using partial determinism information can improve sequentiality of accesses.

Temporal Locality

- There is a substantial overlap in the working sets across days.



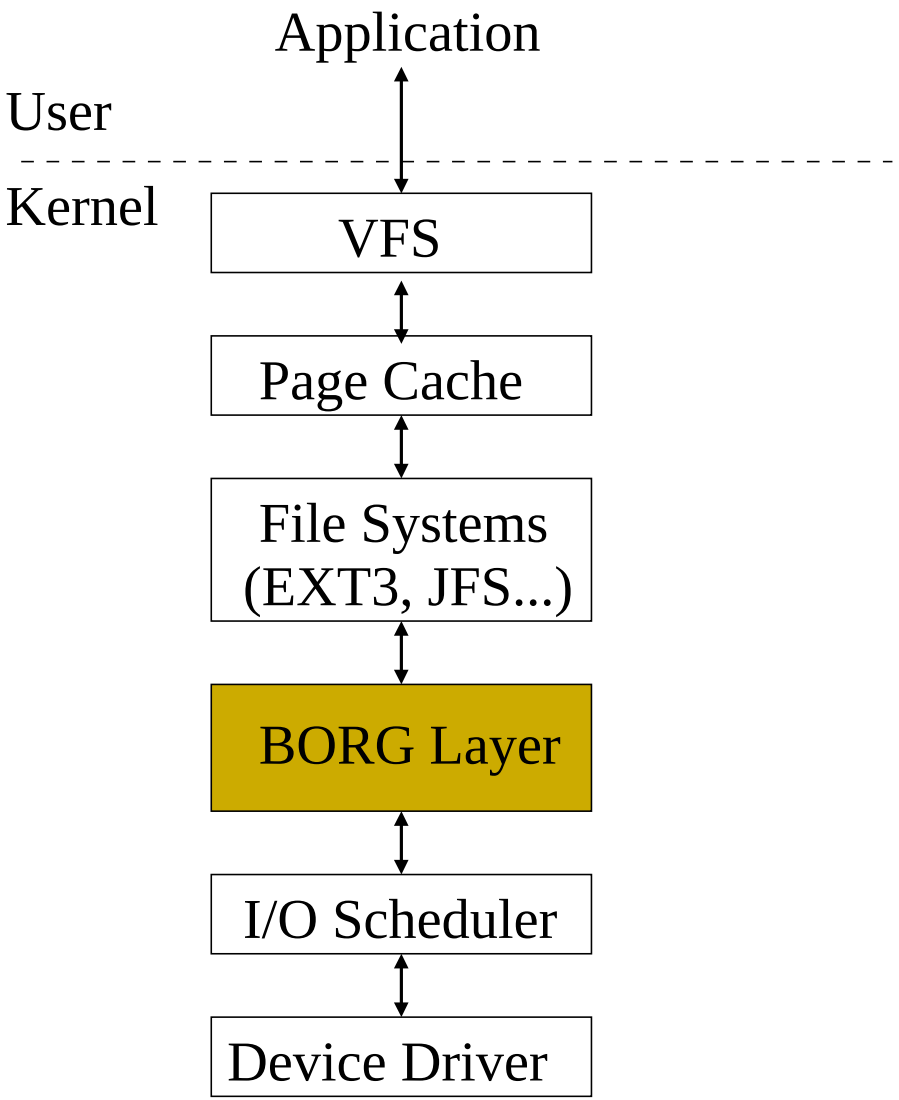
The Opportunity

Using information of past I/O activity for optimizing layout can improve performance.

BORG in a nutshell

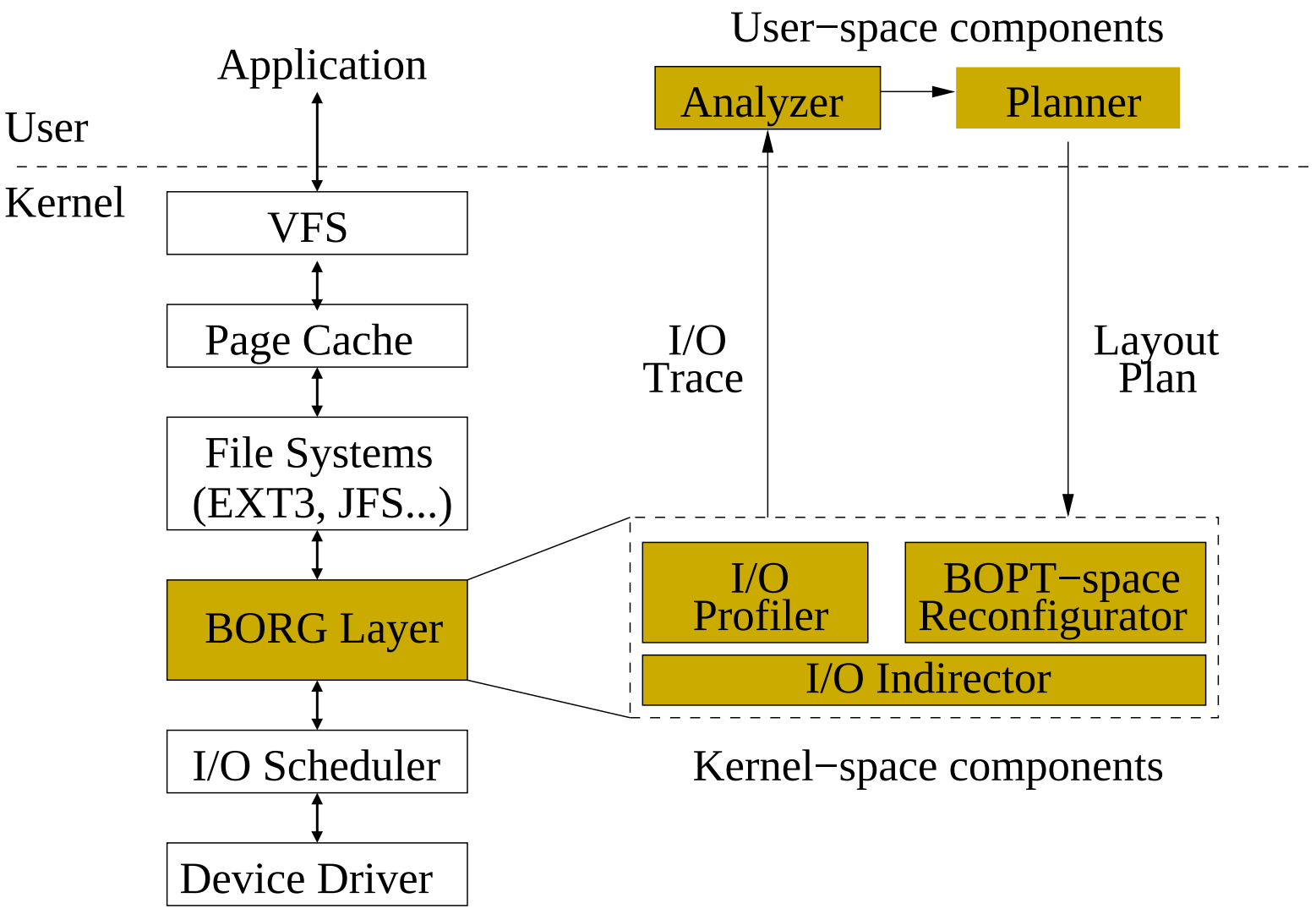
- Uses block access patterns to identify hot block sequences in the workload.
- Reorganizes blocks in a separate *B*ORG *OPT*imized partition (BOPT)
- Assimilates write request in the partition
- Operates in the background
- Can be dynamically inserted or removed when required
- Is independent of filesystems
- Maintains consistency by maintaining a persistent page-level indirection map.

System Architecture

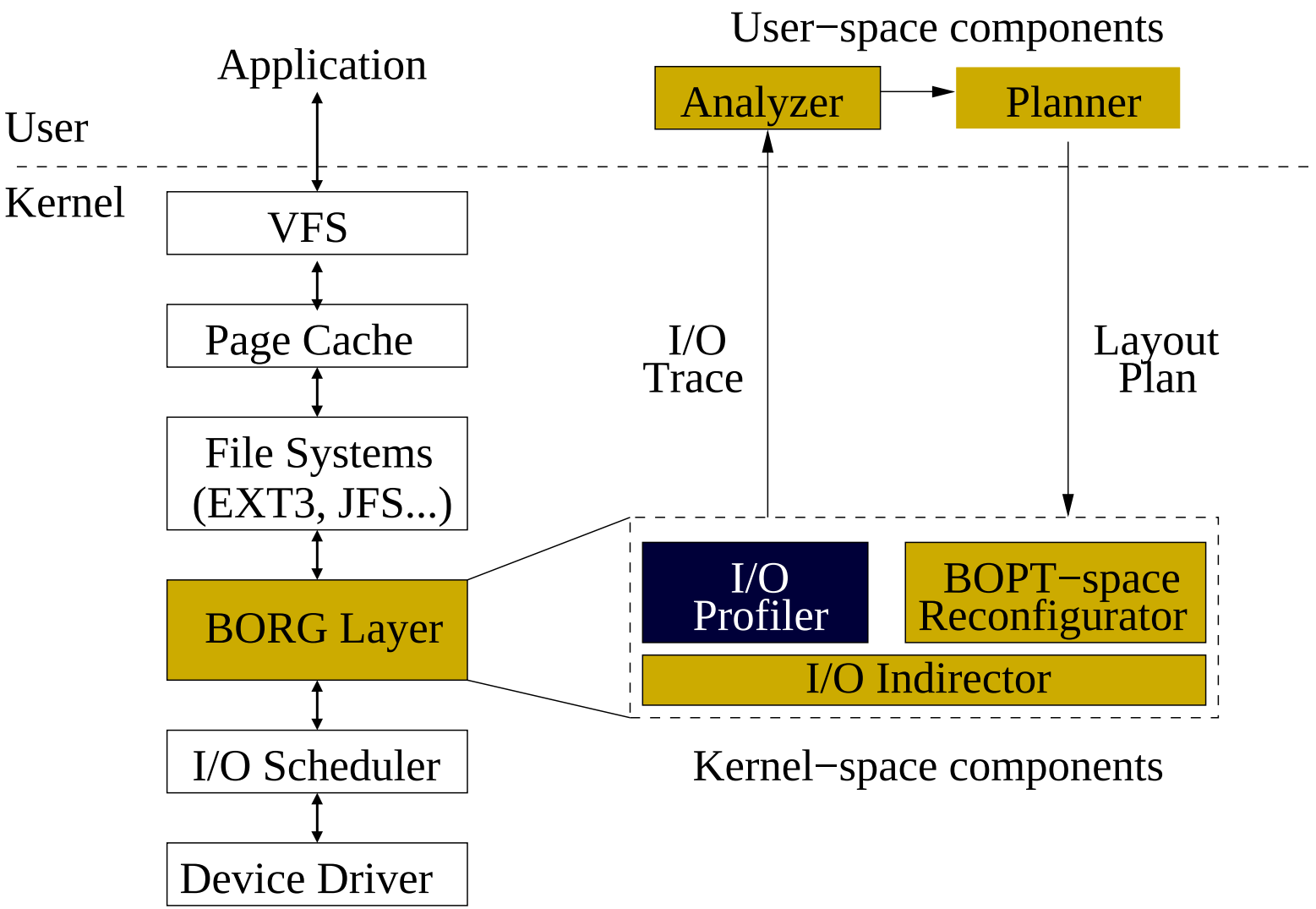


Legend: Existing components New components

System Architecture



System Architecture



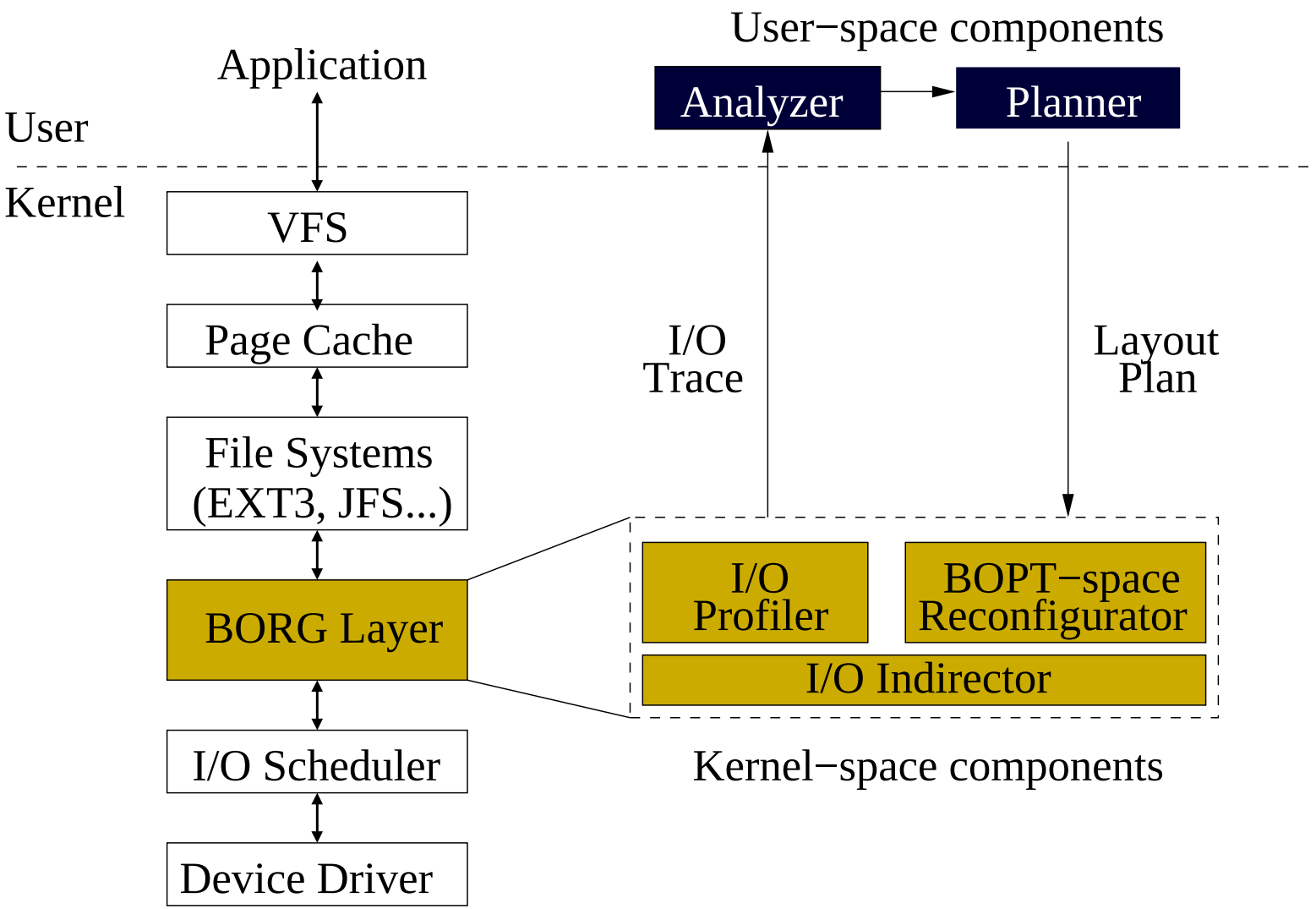
I/O Profiler

- Each I/O operation logged with:
  - ✓ Temporal Attribute: Timestamp
  - ✓ Process-level Attributes: Process ID, name
  - ✓ Block-level attribute: Start LBA, length of I/O, Mode (R/W)

Sample Trace

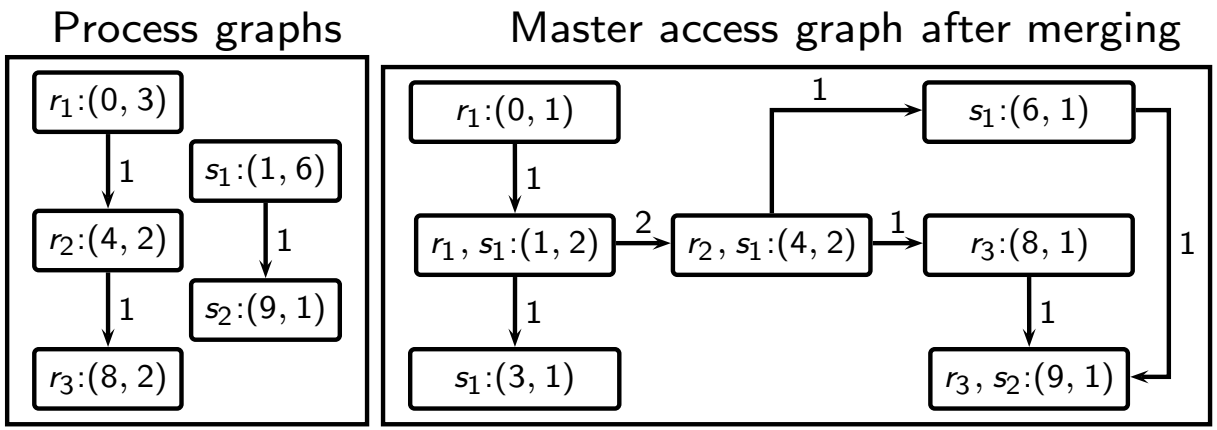
[Timestamp]	[PID]	[Exec.]	[StartLBA]	[Size]	[Mode]
705423195774700	5745	screen	6914207	32	R
705423259644748	5755	utempter	24379775	8	R
705423379492524	5755	utempter	24787567	8	R
705423421266908	5753	bash	7498311	24	R
705423454005104	5755	utempter	24793415	8	R
705423493292648	5753	bash	34543375	64	R
705423565122668	5766	stty	34543439	16	R
...	...	...	...	...	...

System Architecture



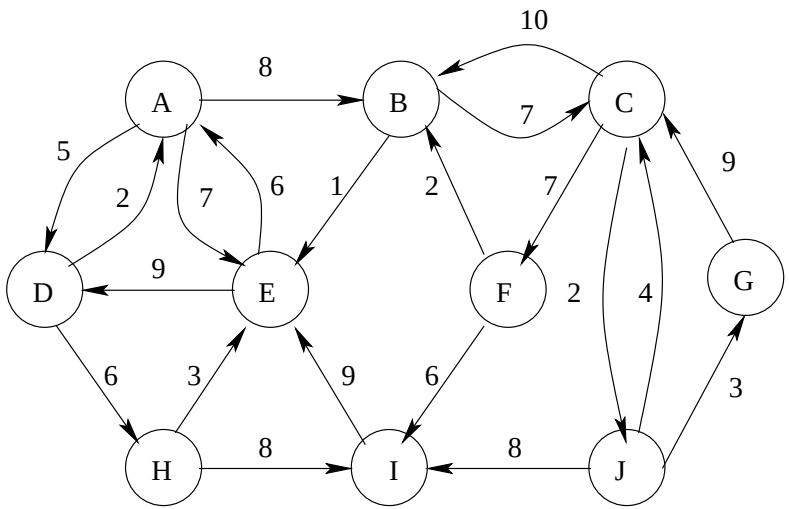
Analyzer

- Builds a per-process directed, weighted graph
- Vertex is the per request LBA range (Start LBA, length)
- Edge is a temporal dependency between two ranges
- Weights represent frequency of access
- Graphs merged into a single master access graph



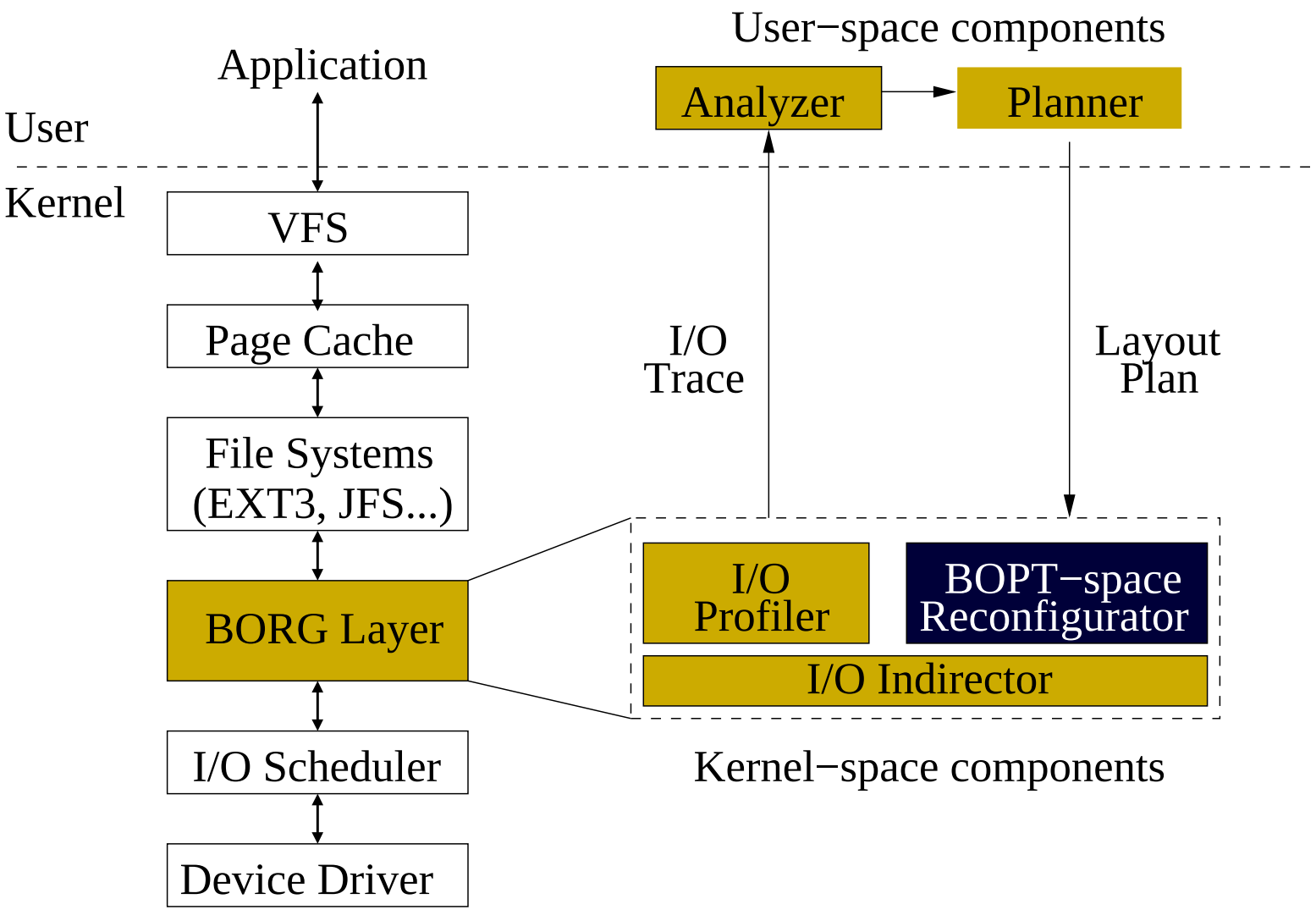
Planner

- Uses master access graph as input
- Chooses the most connected node for initial placement
- Chooses the node most connected to already placed node-set
- Places it depending on its direction of the connecting edge

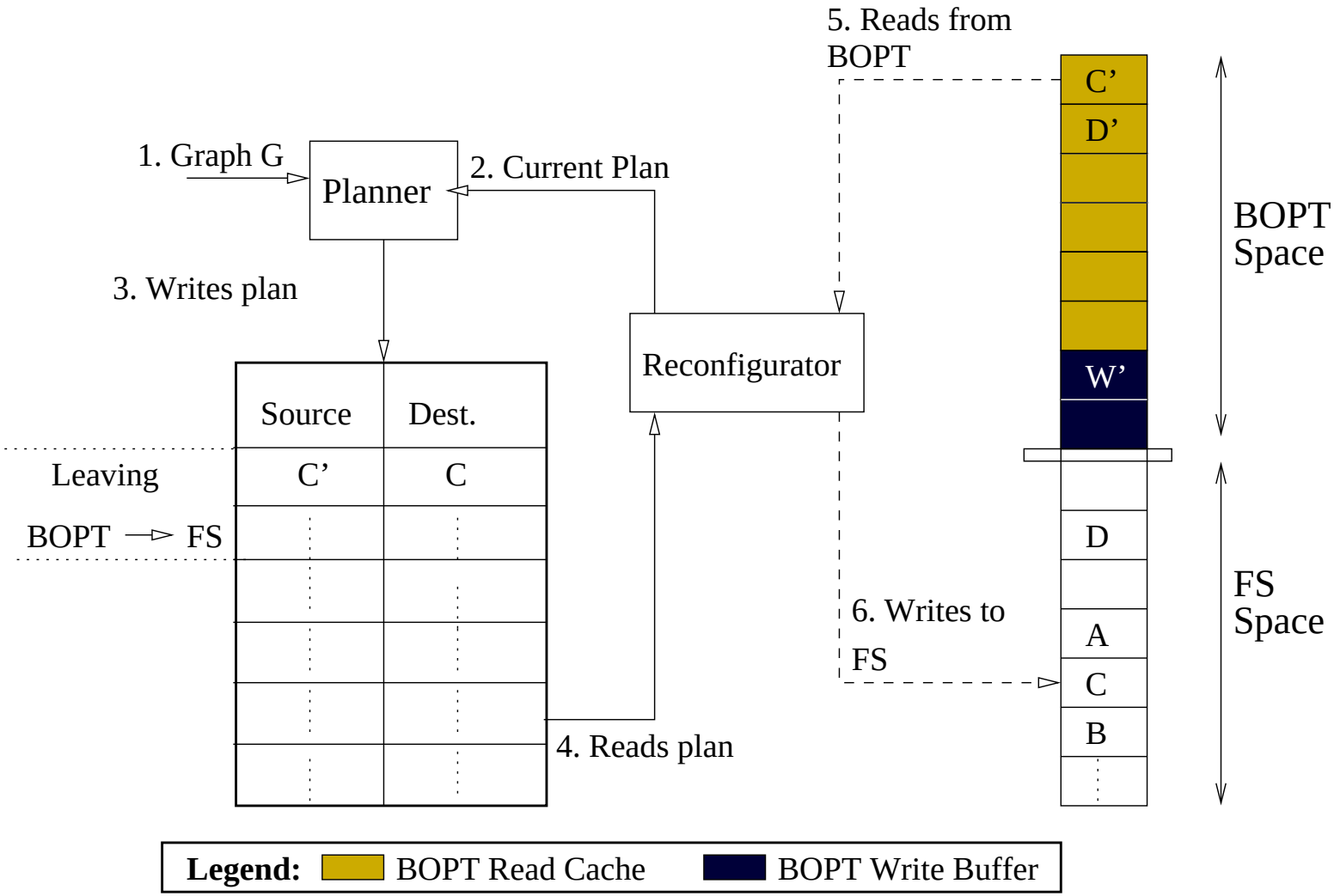


F → H → J → A → G → C → B → E → D

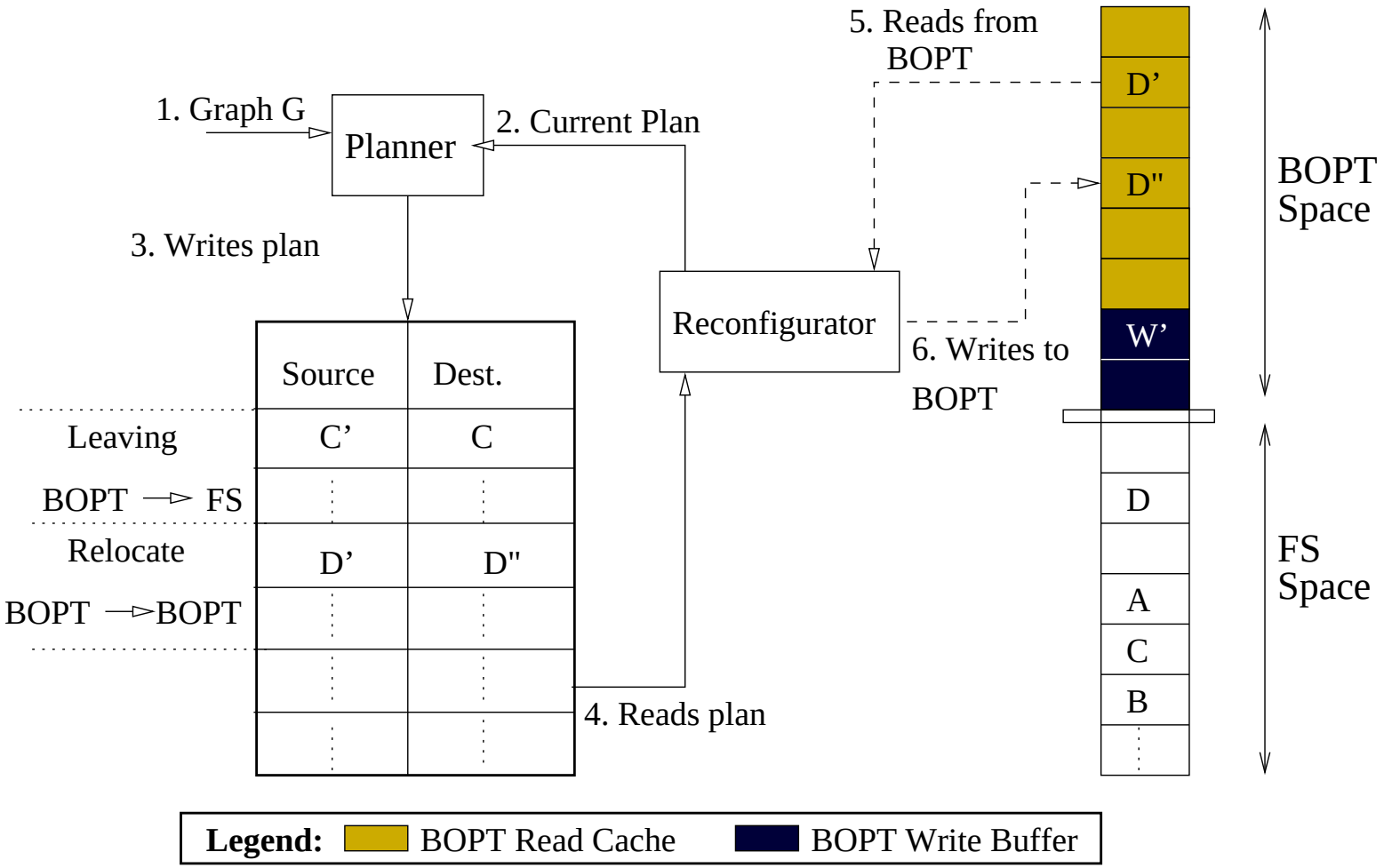
System Architecture



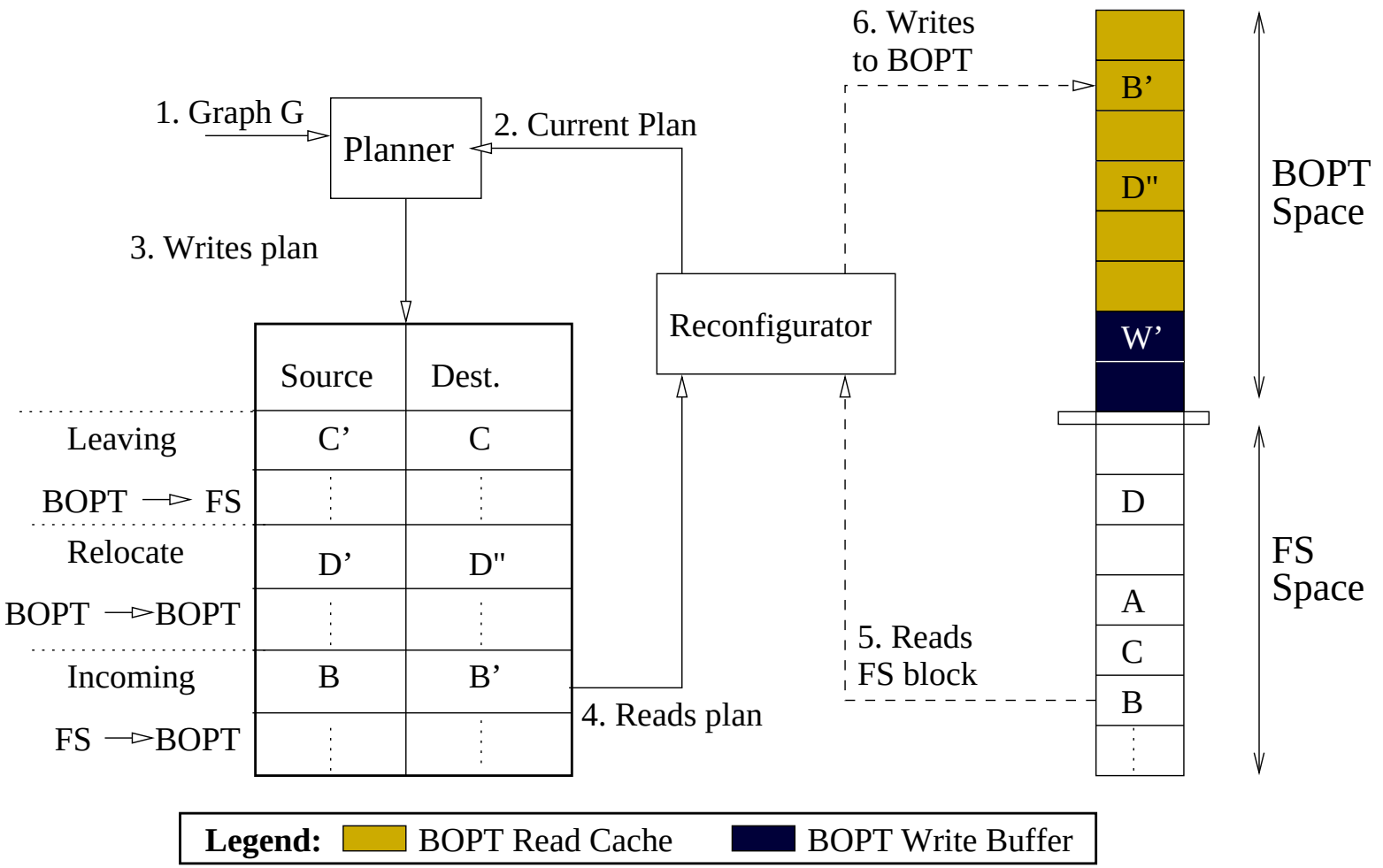
Reconfigurator



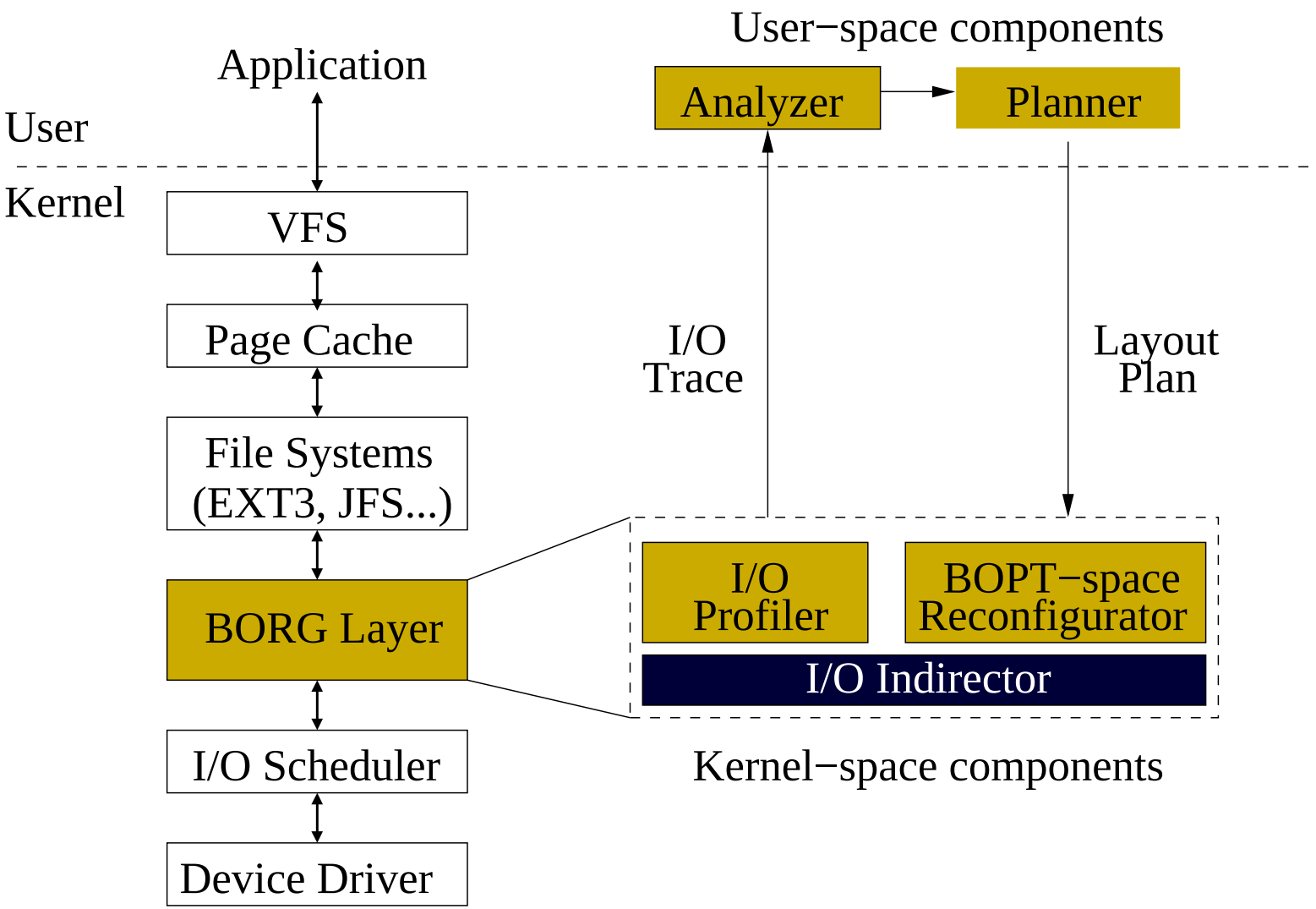
Reconfigurator



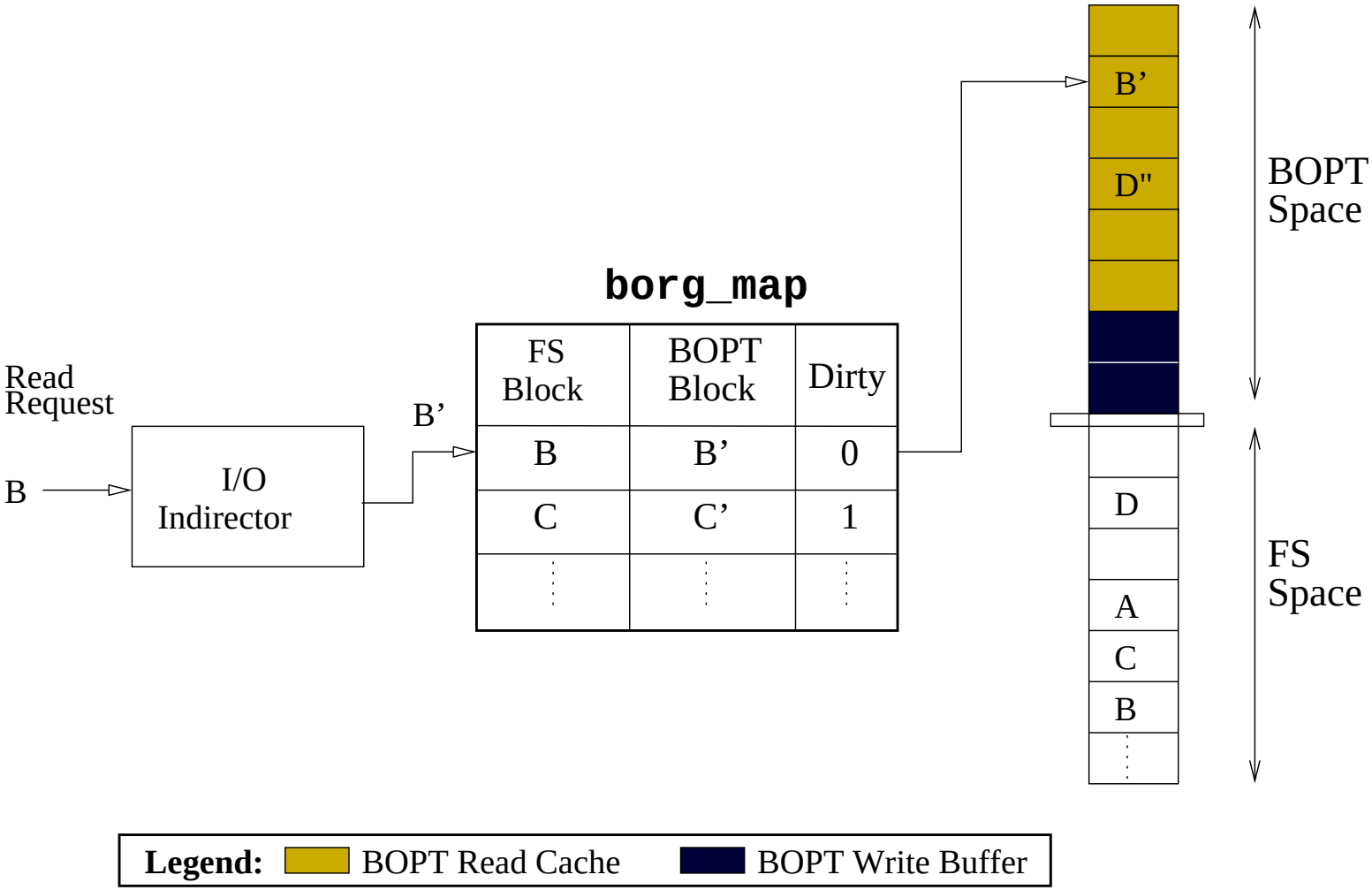
Reconfigurator



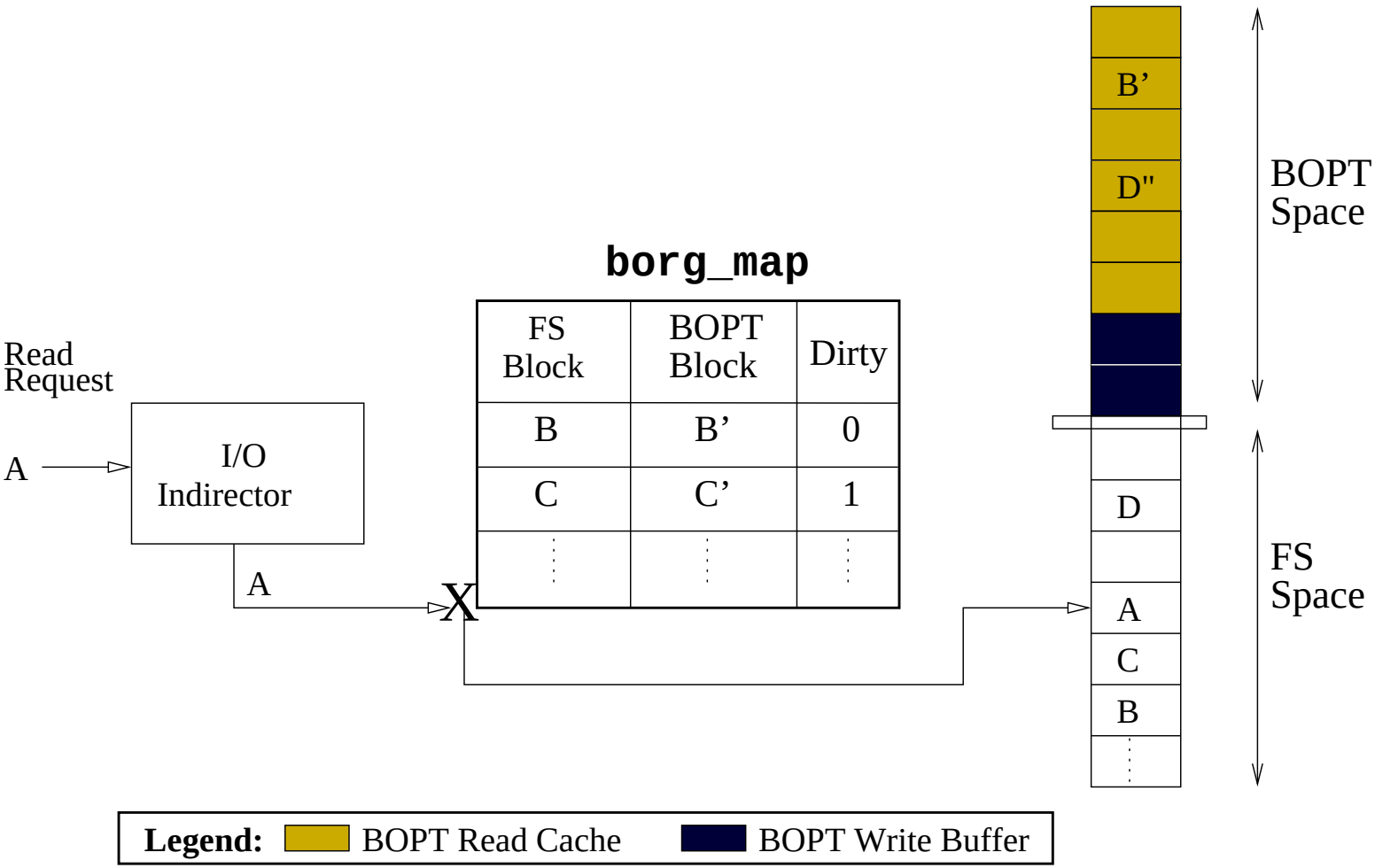
System Architecture



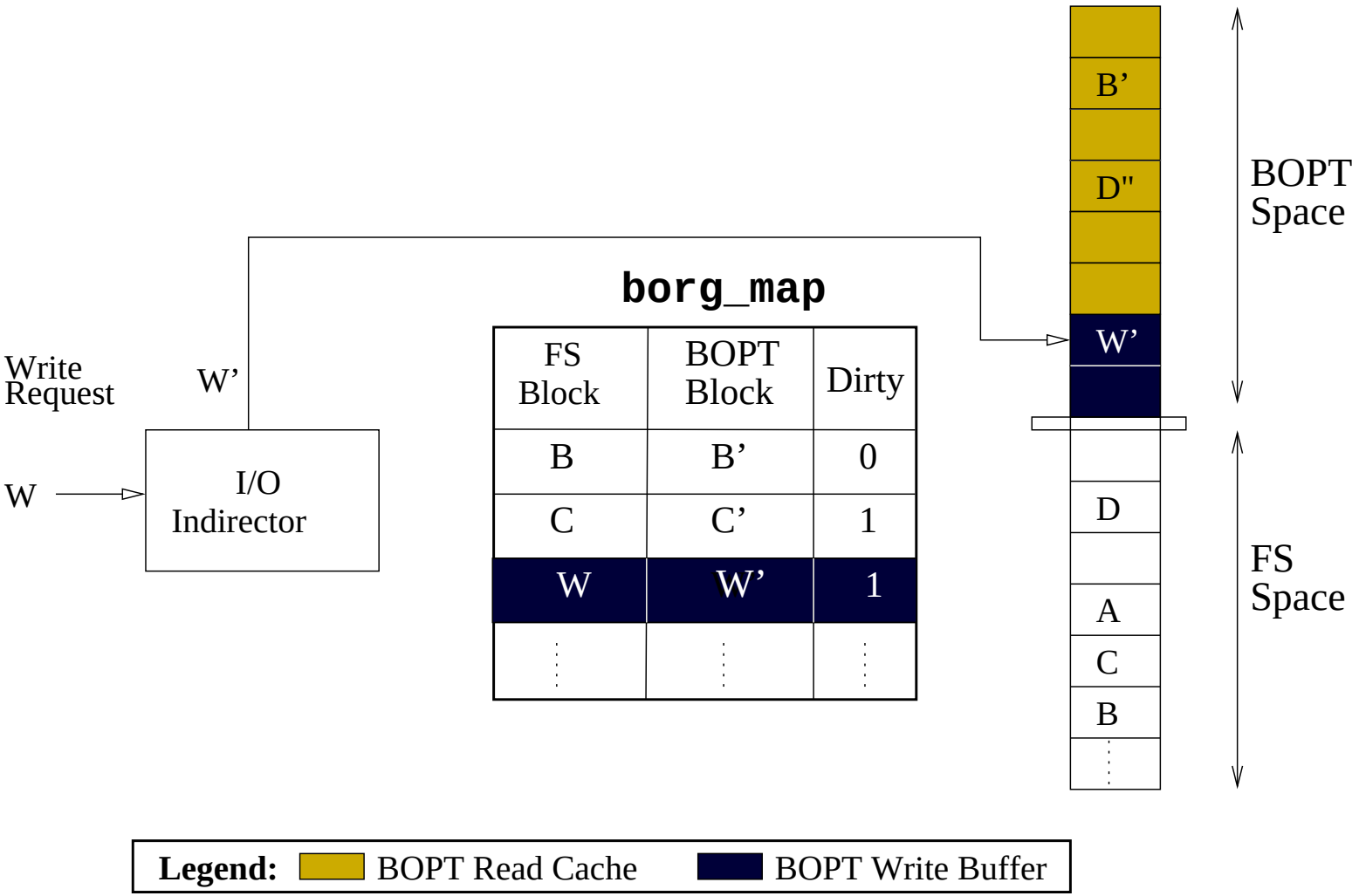
I/O Indirector



I/O Indirector



I/O Indirector



Evaluation

Goals

- How effective is BORG?
- What are the overheads?
- When is it not effective?
- How sensitive is it to different parameters?

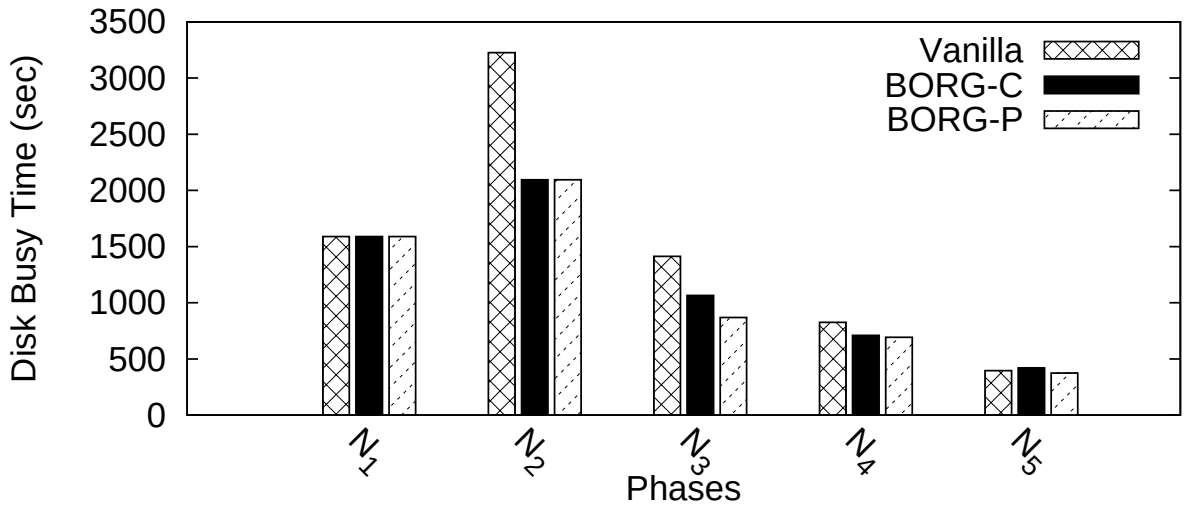
Setup

- Metric - Total disk busy times
- 5 hosts with different configurations
- Linux 2.6.22 kernel
- *reiserfs* and *ext3*

Busy times for Webserver

Setup

- Over 1.1 million requests to over 255,000 files in one week.
- BOPT size 8 GB, 4 Reconfigurations
- Evaluated BORG with *cumulative* and *partial* traces



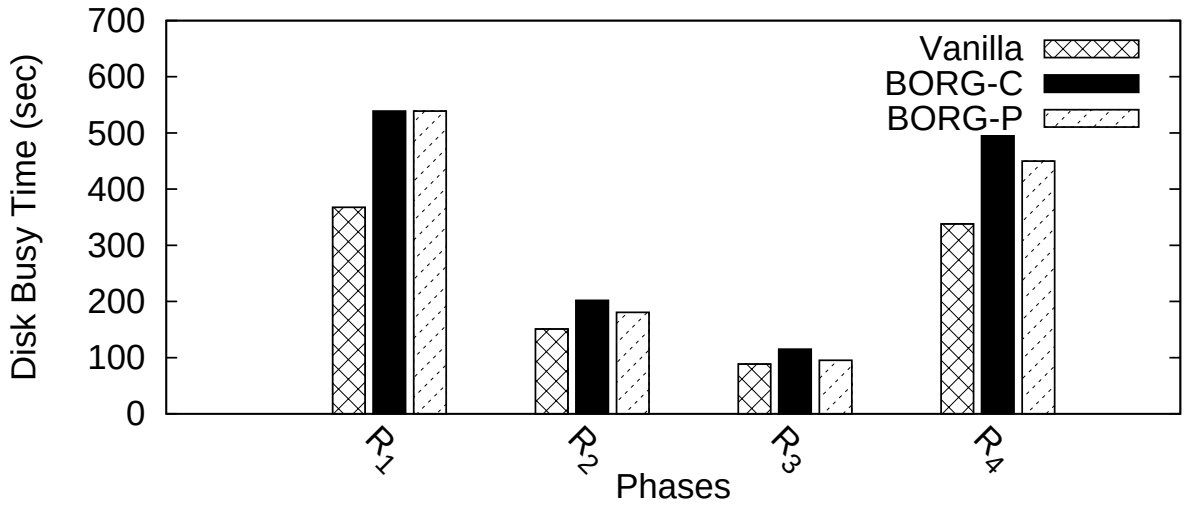
Summary

14-35% reduction in busy times for cumulative and 5-39% for partial traces.

Busy times for Webserver

Setup

- Over 1.1 million requests to over 255,000 files in one week.
- BOPT size 8 GB, 4 Reconfigurations
- Evaluated BORG with *cumulative* and *partial* traces



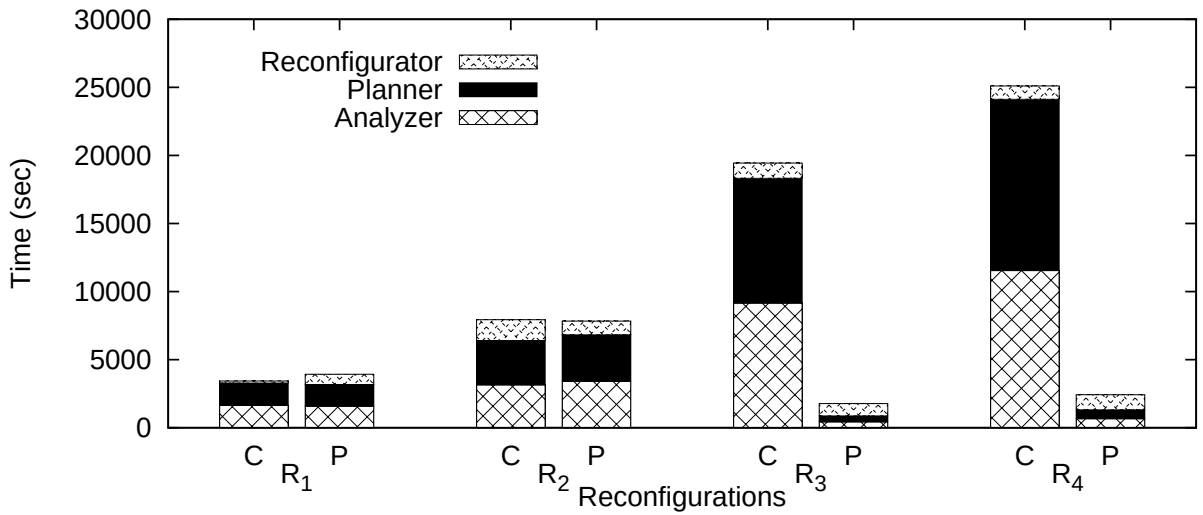
Summary

- Busy times higher in reconfiguration phases due to copy overheads.

BORG Overhead

Setup

- Over 1.1 million requests to over 255,000 files in one week.
- BOPT size 8 GB, 4 Reconfigurations
- *Cumulative* and *partial* traces



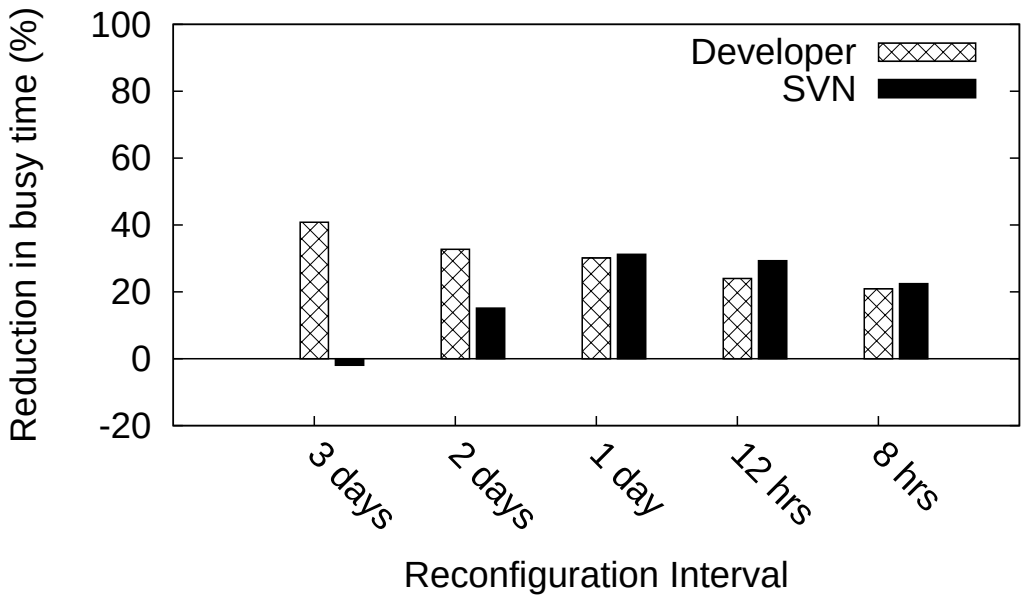
Summary

- Linear increase in planning and analysis overheads for cumulative traces.

Sensitivity Analysis - Reconfiguration Interval

Setup

- Interval 8 hours - 3 days, 1 GB BOPT, with 50% write buffer



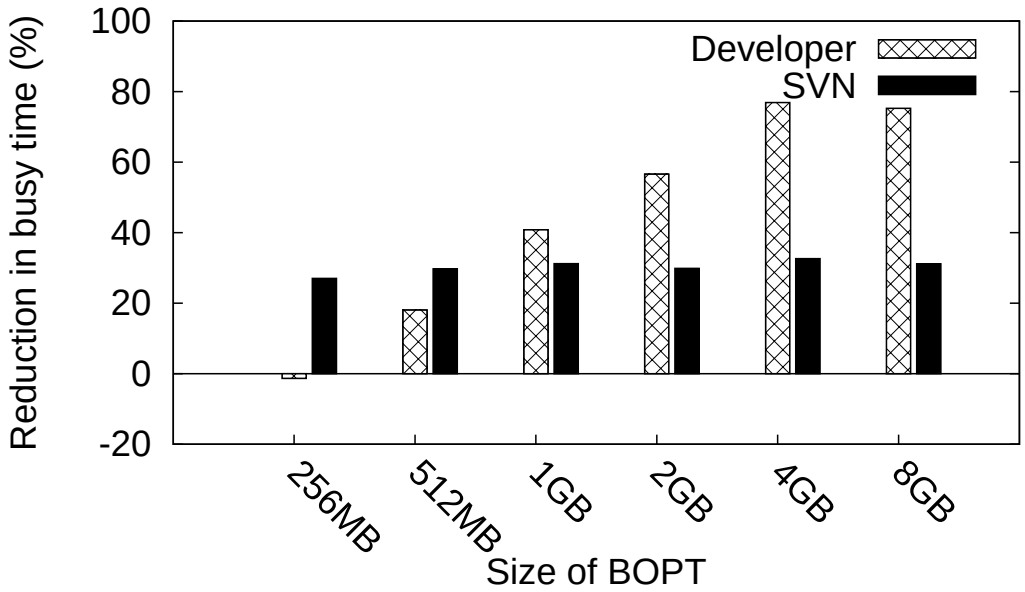
Summary

- Smaller intervals lead to better performance for frequently changing workloads.

Sensitivity Analysis - BOPT Size

Setup

- BOPT size 256 MB - 8 GB, with 50% write buffer



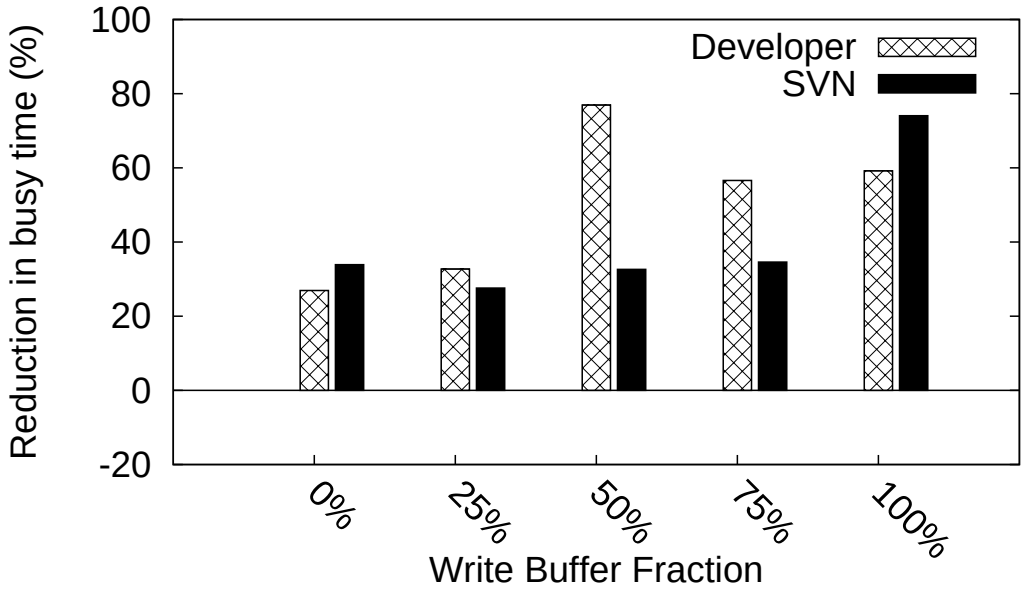
Summary

- Developer: Performance increases with increase in size
- SVN: Improvement is same due to smaller working set size.

Sensitivity Analysis - Write Buffer Size Variation

Setup

- Write buffer 0 - 100%



Summary

- Incorrect size can impact performance

BORG Summary and Future Work

Conclusions

- BORG improves I/O sequentiality and restricts disk head movement
- Disk busy times reduction ranges from 6% to 50% for untuned systems
- Disk busy times can decrease upto 80% with careful tuning
- BORG overheads are within acceptable limits

Future Work

- Exploring alternate layout strategies
- Automated reconfigurations
- Automated configuration of parameters

Thank you!

Related Work

- File System Level Approaches - LFS, PLACE, HFS, FS2
- Block Level Approaches - Cylinder Shuffling, Disk Caching Disk, ALIS