An Efficient Commit Protocol Exploiting Primary-Backup Placement in a Parallel Storage System

Haruo Yokota
Tokyo Institute of Technology

My Research Interests

- Data Engineering + Dependable Systems
  - Dependable Storage Systems
  - Distributed Databases
  - Dependable Web Service Compositions
  - XML Databases
  - …
- In this presentation, focus on
  - Dependable Parallel Storage Systems
    - A Distributed Commit Protocol on the Storage System

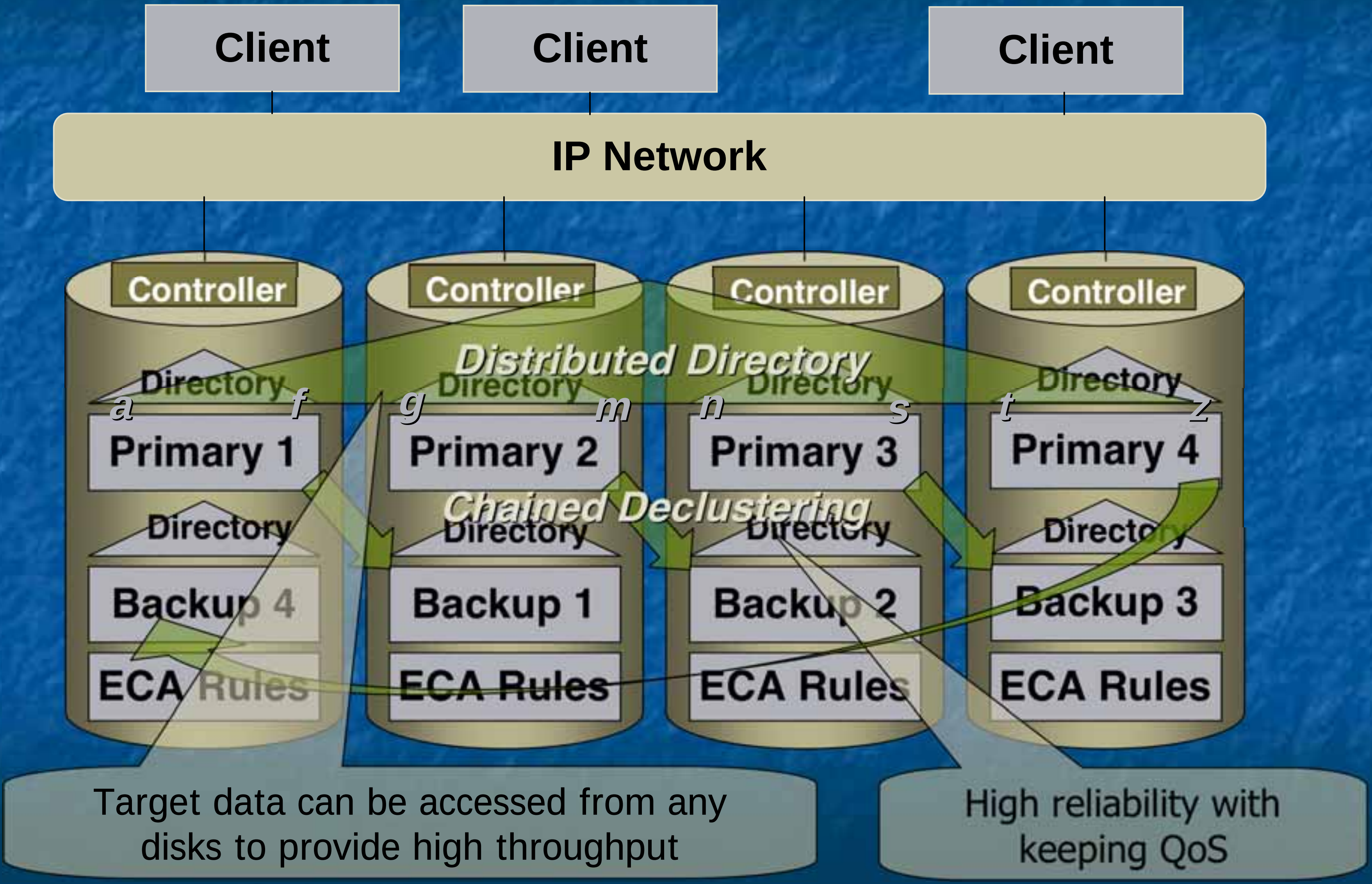
Backgrounds

- Currently, the parity calculation based parallel storage systems, such as RAID3-6, are widely used.
  - However, they have a weakness of drastic performance degradation under failures and recovery processes.
- We choose the primary-backup approach to keep quality of service under failures and recovery.
  - It enables us to store contentious stream contents, such as video streams, in it, also.
- We proposed *Autonomous Disks* [Yokota, 1999]
  - Each disk drive is interconnected in a network to configure a storage cluster, and equipped with a disk-resident intelligent processor to handle data distribution, load balance and disk failures.
  - In the cluster, data are partitioned into all member disks.

Key Technologies of AD

- Access Methods
  - We proposed the *Fat-Btree* as a distributed directory to access data distributed in the disk cluster [Yokota, ICDE1999]
- Data Placement (load/amount balance)
  - We first adopt *Chained Declustering* method [Hsiao and Dewitt, 1990] for locating the primary and backup
  - Then we propose *Adaptive Overlapped Declustering* method to balance both access load and data amount of each disk [Watanabe and Yokota, ICDE2005]

Configuration of an AD Cluster

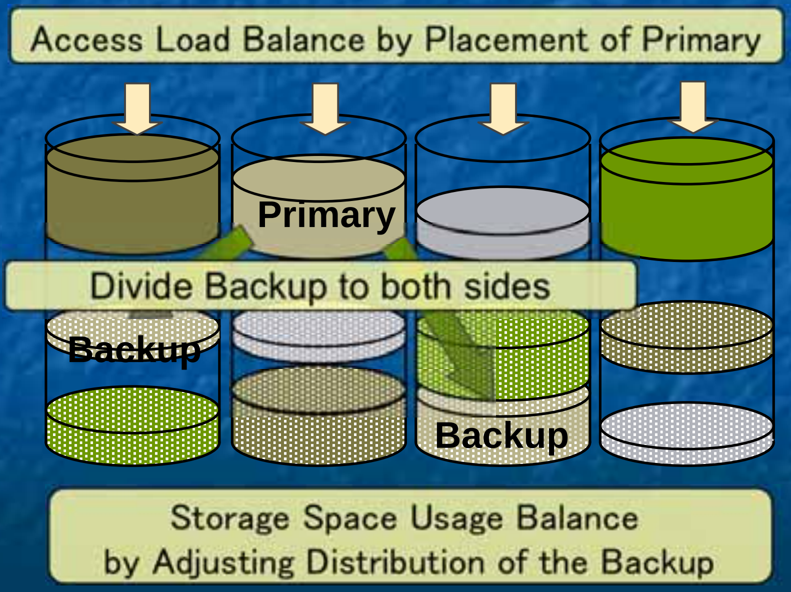


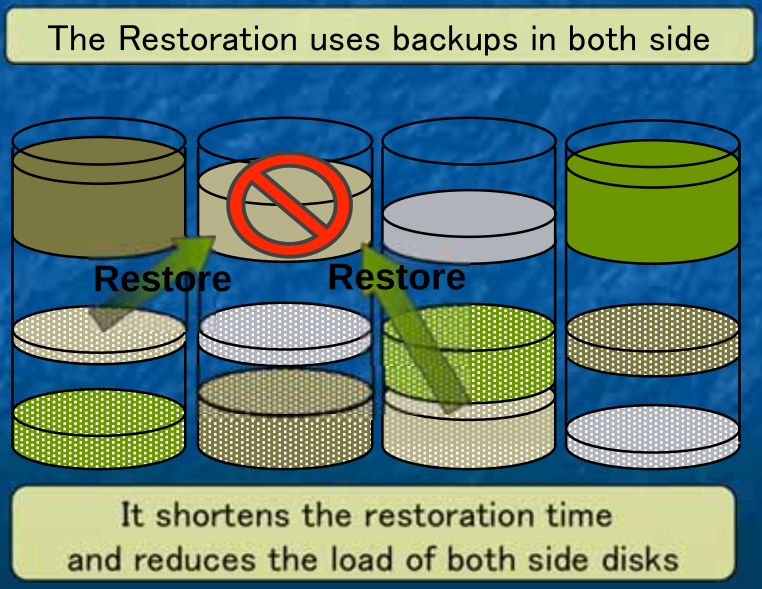
Target data can be accessed from any disks to provide high throughput

High reliability with keeping QoS

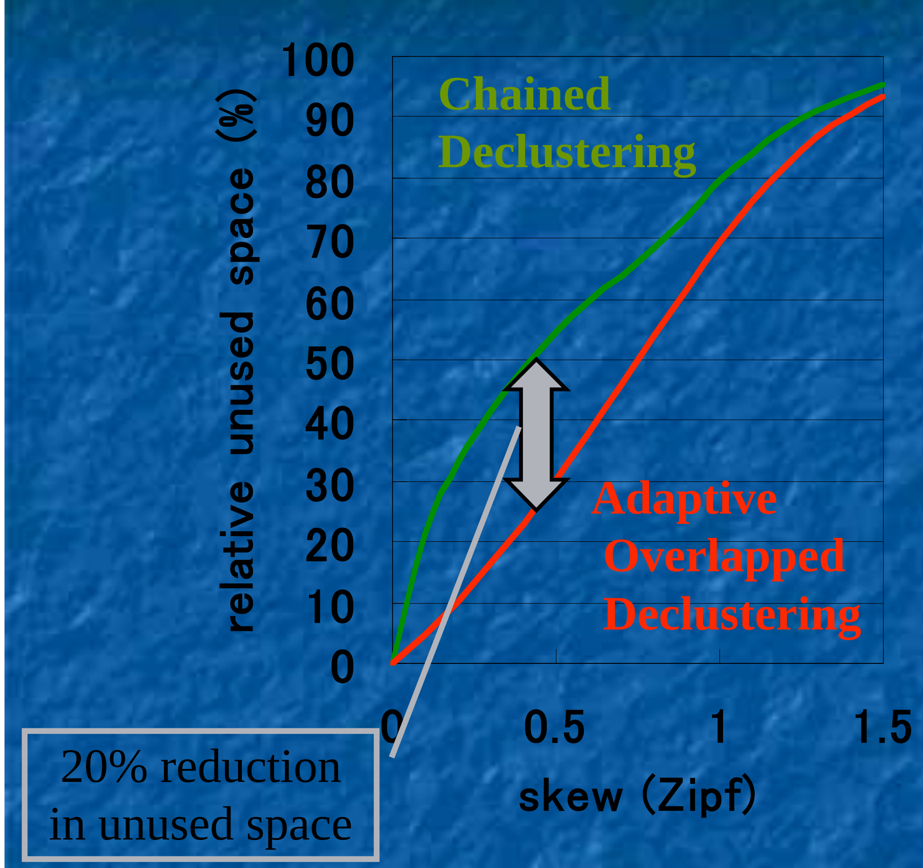
AOD: Adaptive Overlapped Declustering

- It aims to balance both access load and space utilization
  - We distribute the backup data to both sides of the primary data with assuming that the backup data is not accessed usually
  - Access load is balanced by only the placement of primary data.
  - Space utilization is balanced by adjusting the backup distribution.
- AOD is also effective to shorten the recovery time because it uses both side backups to restore the primary.

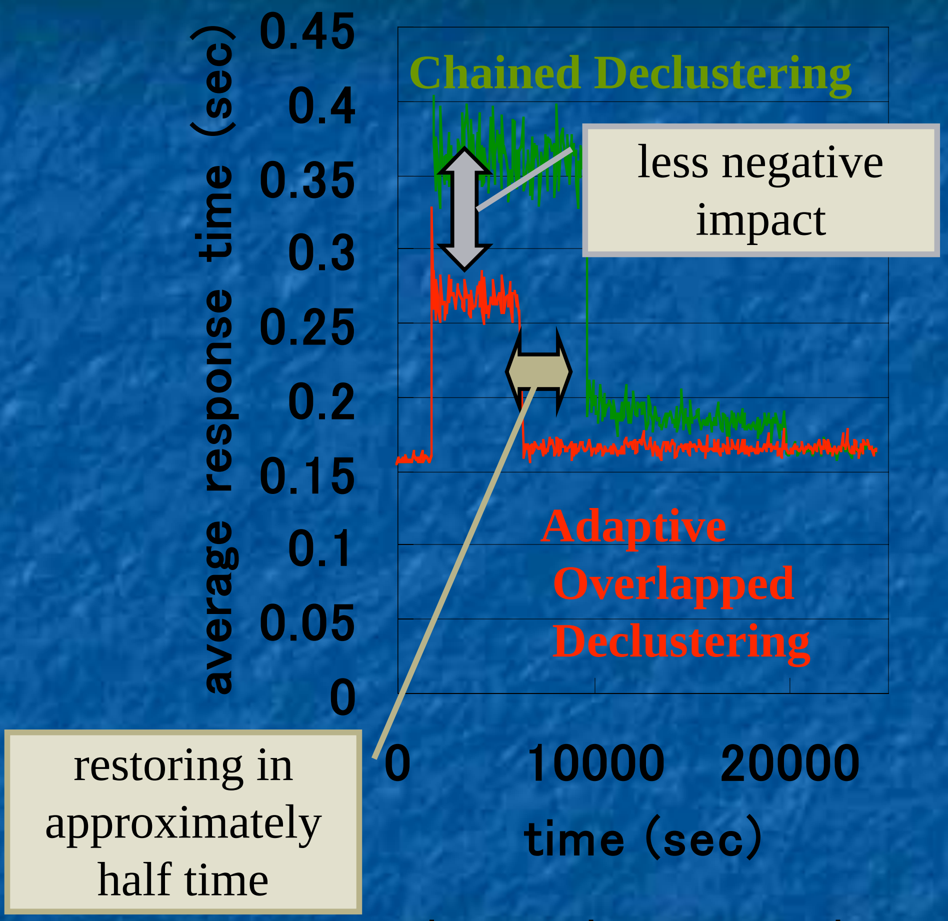

Access Load Balance by Placement of Primary
Primary
Divide Backup to both sides
Backup
Backup
Storage Space Usage Balance by Adjusting Distribution of the Backup


The Restoration uses backups in both side
Restore
Restore
It shortens the restoration time and reduces the load of both side disks

Simulation Results



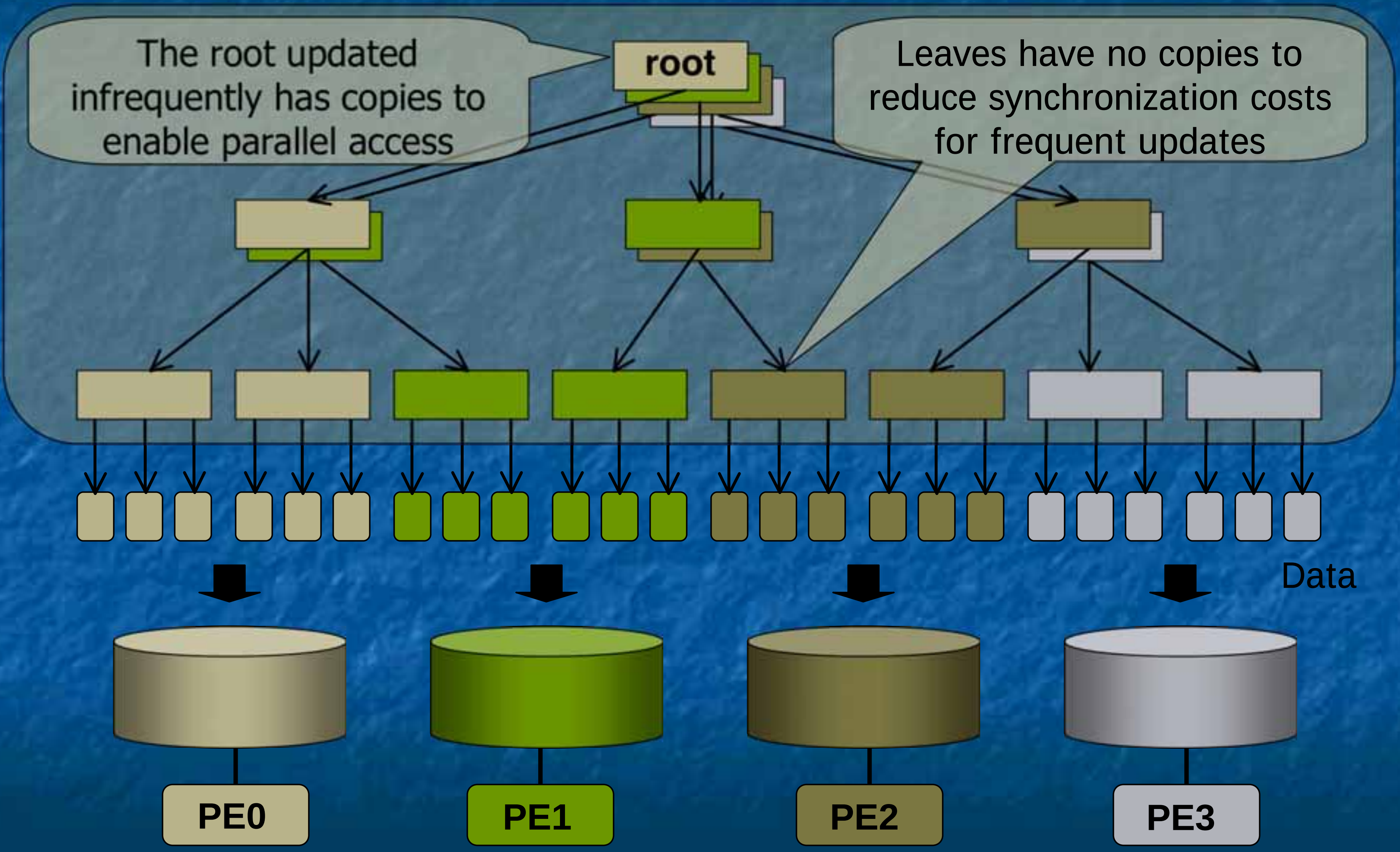
Unused space



Recovery time and response time

- The graphs illustrate that the effect of balancing load and amount, and of shortening the recovery time

A Fat-Btree


The root updated infrequently has copies to enable parallel access
root
Leaves have no copies to reduce synchronization costs for frequent updates
Data
PE0
PE1
PE2
PE3

Transaction Control on Fat-Btree

- The strategy of partial duplication adds the complexity to structure modification operations (SMOs), such as page splits and merges, since the index pages are partially copied across PEs.
  - When an SMO occurs at an index page, all PEs having a copy of the index page must be updated synchronously.
- We developed concurrency control protocols suited for Fat-Btree [Miyazaki et al. 2002, Yoshihara et al. 2006].
- Atomic Commit Protocol (ACP) becomes imperative to guarantee the correctness of data.

BA-1.5PC

- We propose a new atomic commit protocol suited for distributed storage adopting primary backup data placement under the fail-stop failure model.
    - It has a low-overhead log mechanism that eliminates the blocking disk I/O (*async-nWAL*).
    - It removes the voting phase from commit processing to gain a faster commit process.
- We named it *Backup-Assist 1.5-Phase Commit*, or *BA-1.5PC* protocol.

Other Commit Protocols (1/3)

- Two-Phase Commit (2PC) protocol is the best known of all commit protocols for distributed environment.
  - It can produce severe delays while synchronizing the copies under the primary-backup scheme.
  - It can also prone to blocking cohorts if a master fails in the decision phase.
- Many attempts at ACP ameliorate the drawbacks of 2PC

Other Commit Protocols (2/3)

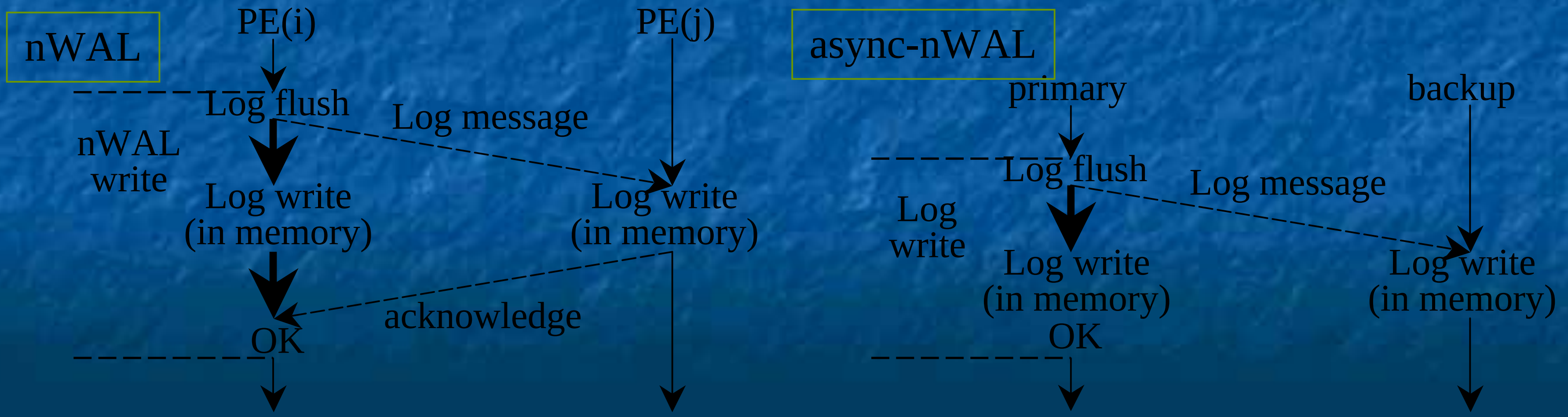
- Low-overhead commit protocols
  - Presumed Abort (PA) protocol, Presumed Commit (PC) protocol [Mohan et al. 1986] reduce number of message exchanges as well as forced log writes
  - Early Prepare (EP) protocol [Stamos et al. 1990] heavy delay at transaction processing
  - Coordinator Log (CL) protocol [Stamos et al. 1993], Implicit Yes Vote (IYV) protocol [Al-Houmaily et al. 1995] concentrate cohorts' log to the master and hamper the system's scalability
  - OPTimistic (OPT) commit protocol [Gupta et al. 1997] reduces lock waiting time by lending the locks holding by transactions in commit processing, but must be carefully designed to avoid cascaded aborting

Other Commit Protocols (3/3)

- Non-blocking commit protocols
  - Three-Phase Commit (3PC) protocol [Skeen 1982] has an extra "buffer phase" between the voting phase and the decision phase suffers severe delay
  - ACP based on Uniform Timed Reliable Broadcast (ACP-UTRB) [Babaoglu et al. 1993], Non-Blocking Single-Phase Atomic Commit (NB-SPAC) protocol [Abdallah et al. 1998] impose a strong assumption "uniform broadcast" upon underlying communication model, it assumes the strong assumption and the cost of message delivery depends on number of participants compromise scalability

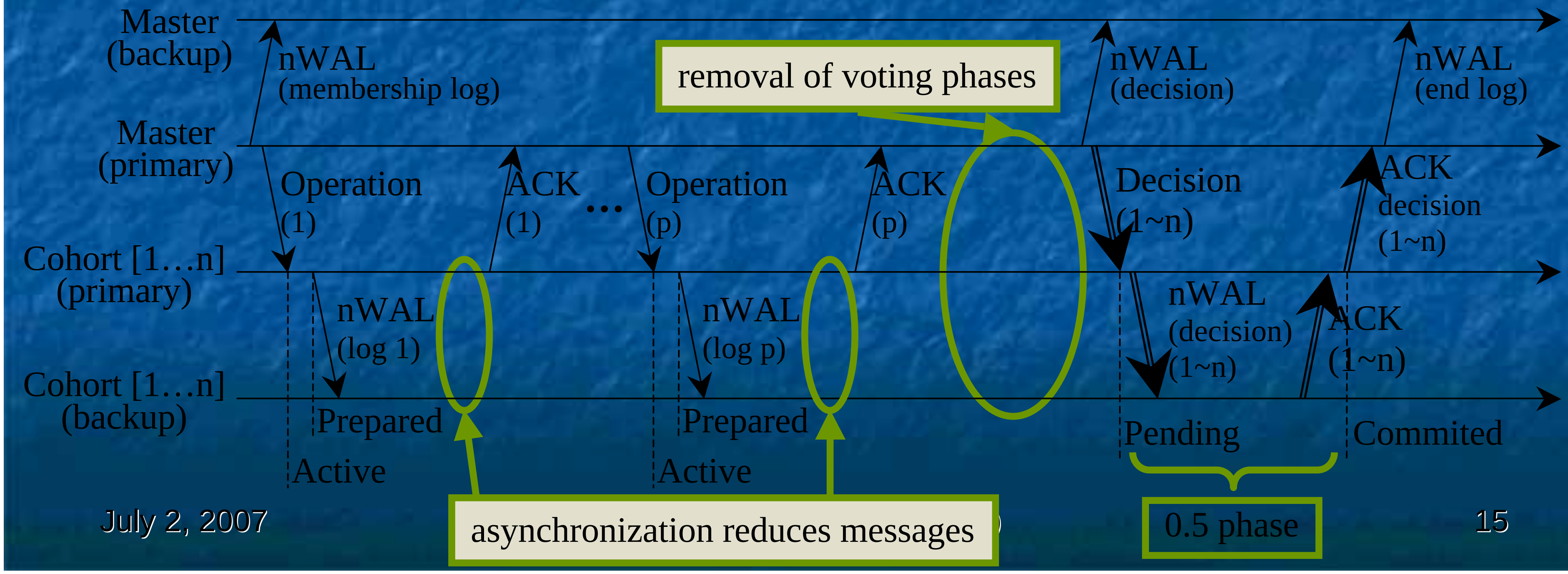
Async-nWAL

- Write-Ahead Log [Härder et al. 1983] (WAL) is widely accepted as a means for providing atomicity and durability in DBs
- Neighbor WAL [Hvasshovd et al. 1996] (nWAL) reduces the overhead of logging of WAL in parallel databases writing the logs into memory of a neighboring PE.
- We propose *Asynchronous-nWAL* (*async-nWAL*)
  - It does not wait for the ACK from neighbor at a log flush operation by deferring the synchronization until the final decision phase of the transaction.



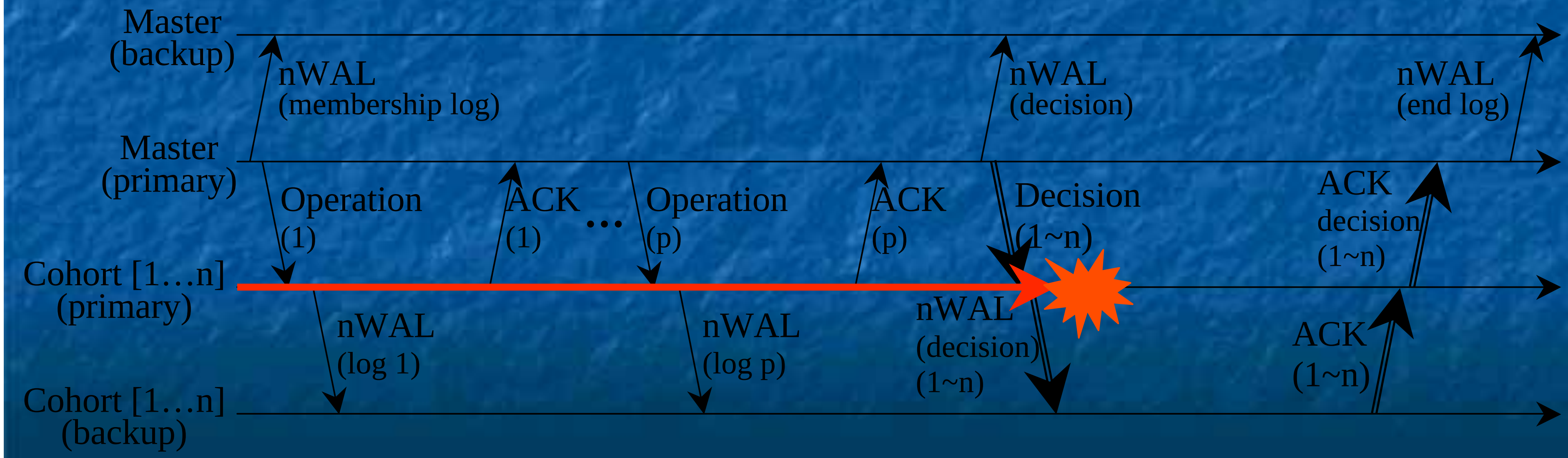
BA-1.5 Phase Commit Protocol

- It removes the voting phase by implicitly including cohorts' votes into the ACK of each operations
- Pending phase
  - A primary cohort waits for the backup at decision phase
    - The only point at which the primary and backup are synchronized
  - A pending phase with a result that can be discarded is counted as 0.5 phase in the commit process
    - The origin of the name 1.5-phase



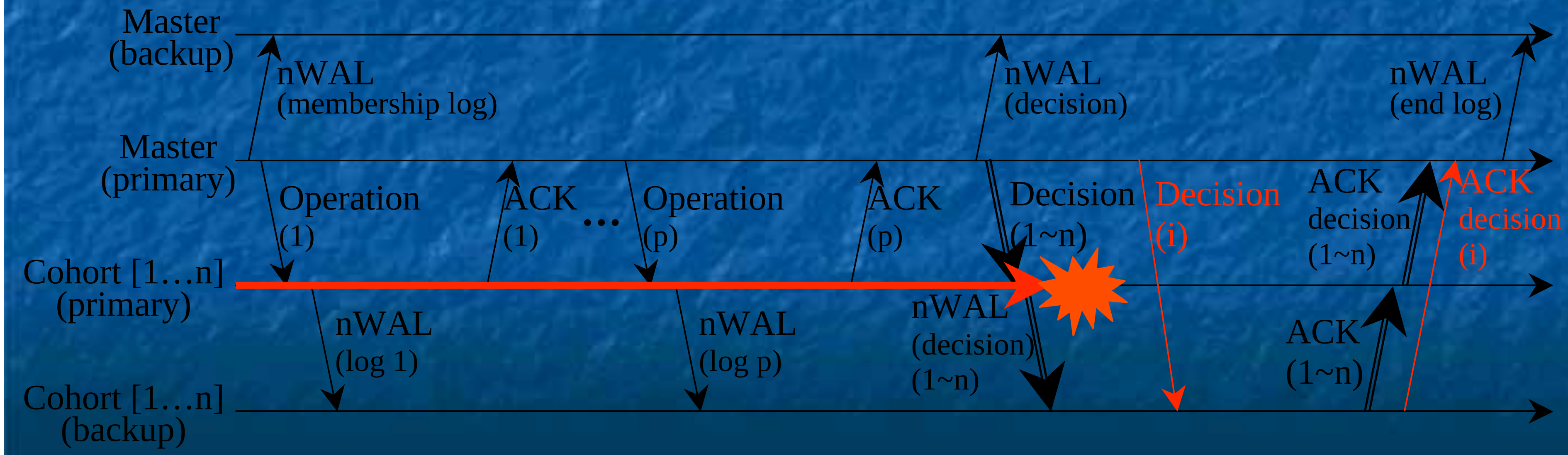
Failure Recovery Strategy (1/2)

- BA-1.5PC handles failures effectively by exploiting the primary-backup structure
- For cohort failures,
  - The backup of that cohort can assume primary cohort's role
    - The backup cohort will maintain log records for operations until the failed cohort is fixed and restarts
    - It can easily restore its state by reading the complete list of log records at its backup and applying them to bring its data up to date



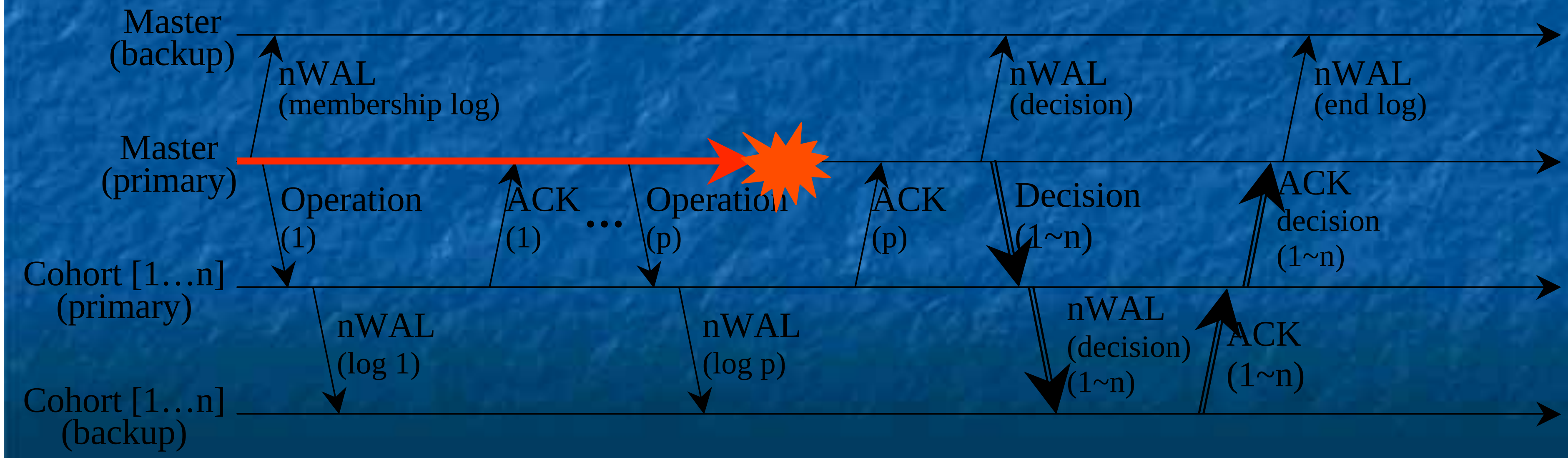
Failure Recovery Strategy (1/2)

- BA-1.5PC handles failures effectively by exploiting the primary-backup structure
- For cohort failures,
  - The backup of that cohort can assume primary cohort's role
    - The backup cohort will maintain log records for operations until the failed cohort is fixed and restarts
    - It can easily restore its state by reading the complete list of log records at its backup and applying them to bring its data up to date



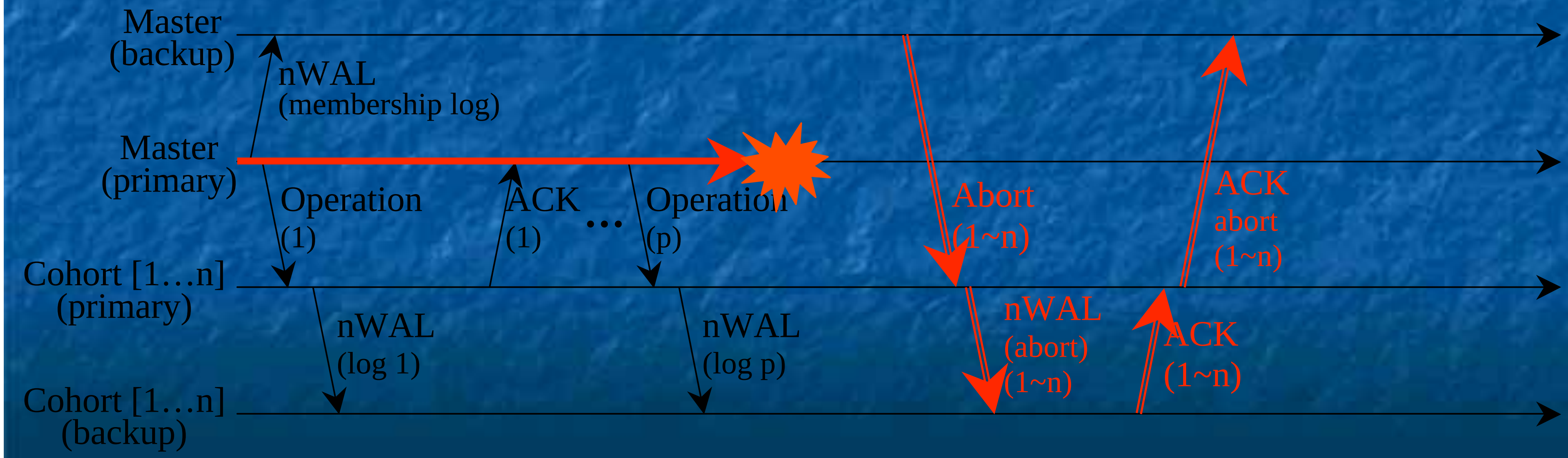
Failure Recovery Strategy (2/2)

- For master failure,
  - It is surmounted by its backup located in its logical neighbor
  - The primary master failure before the decision phase
    - The backup master will automatically abort the transaction
  - The primary master failure during the decision phase
    - The backup master will re-inform all cohorts of the result of transaction, instead of forcing other cohorts to wait until the failed master finishes its restart and recovery



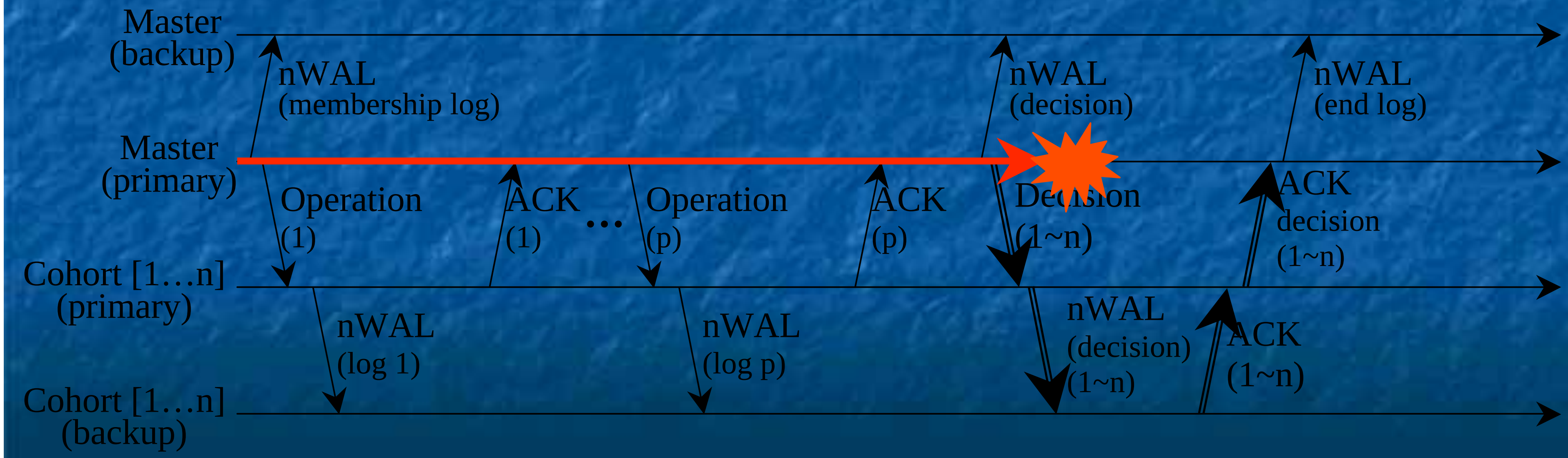
Failure Recovery Strategy (2/2)

- For master failure,
  - It is surmounted by its backup located in its logical neighbor
  - The primary master failure before the decision phase
    - The backup master will automatically abort the transaction
  - The primary master failure during the decision phase
    - The backup master will re-inform all cohorts of the result of transaction, instead of forcing other cohorts to wait until the failed master finishes its restart and recovery



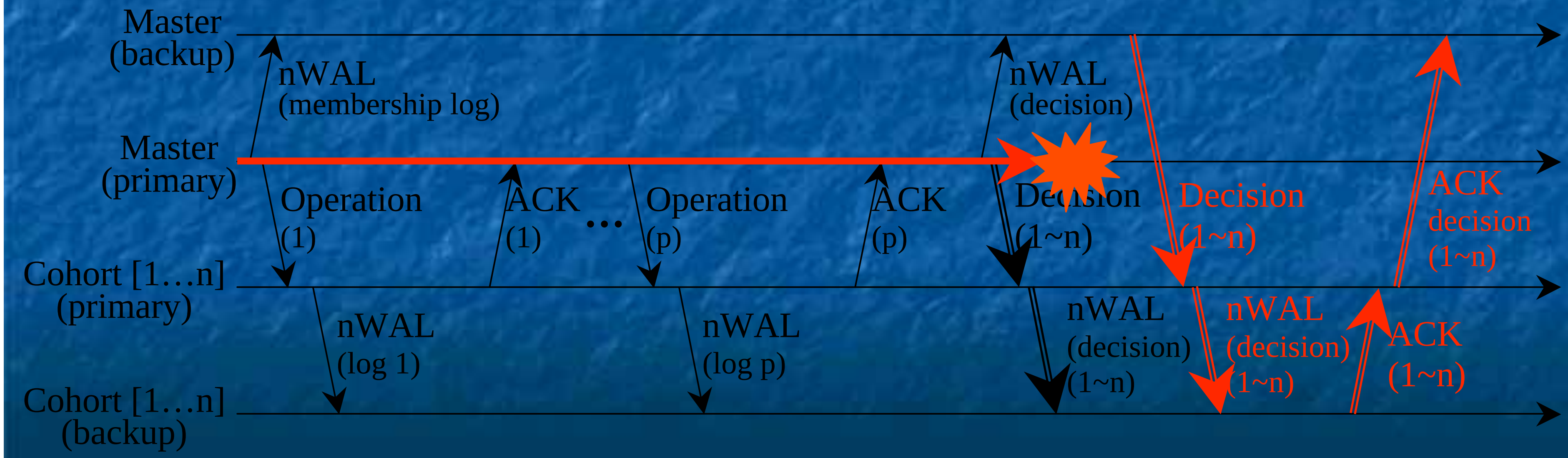
Failure Recovery Strategy (2/2)

- For master failure,
  - It is surmounted by its backup located in its logical neighbor
  - The primary master failure before the decision phase
    - The backup master will automatically abort the transaction
  - The primary master failure during the decision phase
    - The backup master will re-inform all cohorts of the result of transaction, instead of forcing other cohorts to wait until the failed master finishes its restart and recovery



Failure Recovery Strategy (2/2)

- For master failure,
  - It is surmounted by its backup located in its logical neighbor
  - The primary master failure before the decision phase
    - The backup master will automatically abort the transaction
  - The primary master failure during the decision phase
    - The backup master will re-inform all cohorts of the result of transaction, instead of forcing other cohorts to wait until the failed master finishes its restart and recovery



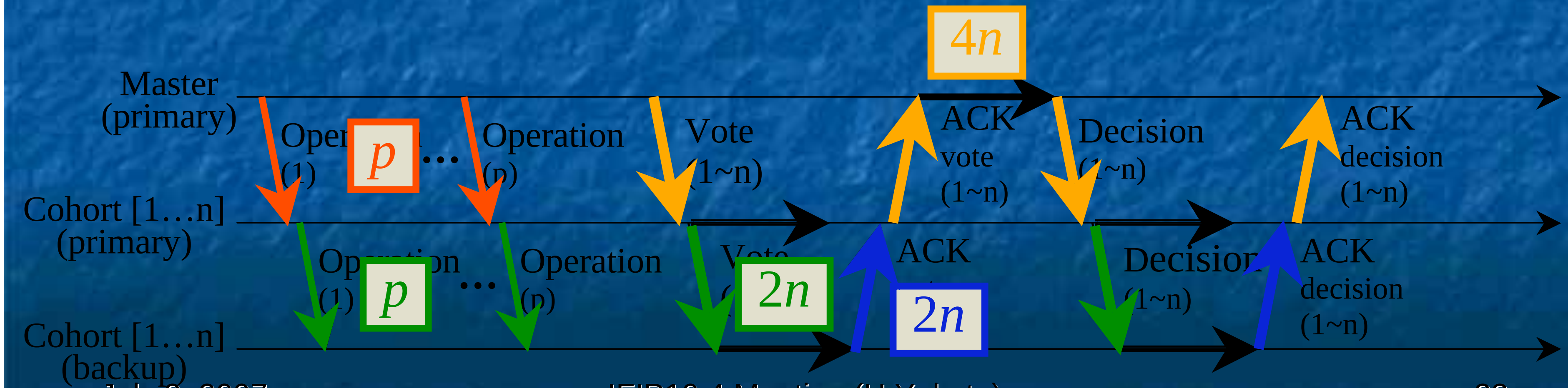
Estimation of Protocol Overheads

- Compare overheads referred to message exchanges and forced disk writes in both transaction processing and commit processing
  - Comparison targets: conventional protocols
    - 2PC, EP
  - Assumption
    - There are *n* cohorts in a transaction besides the master
    - A transaction has *p* operations that are delivered to cohorts

Overheads of 2PC

n: No. of cohorts
p: No. of operations

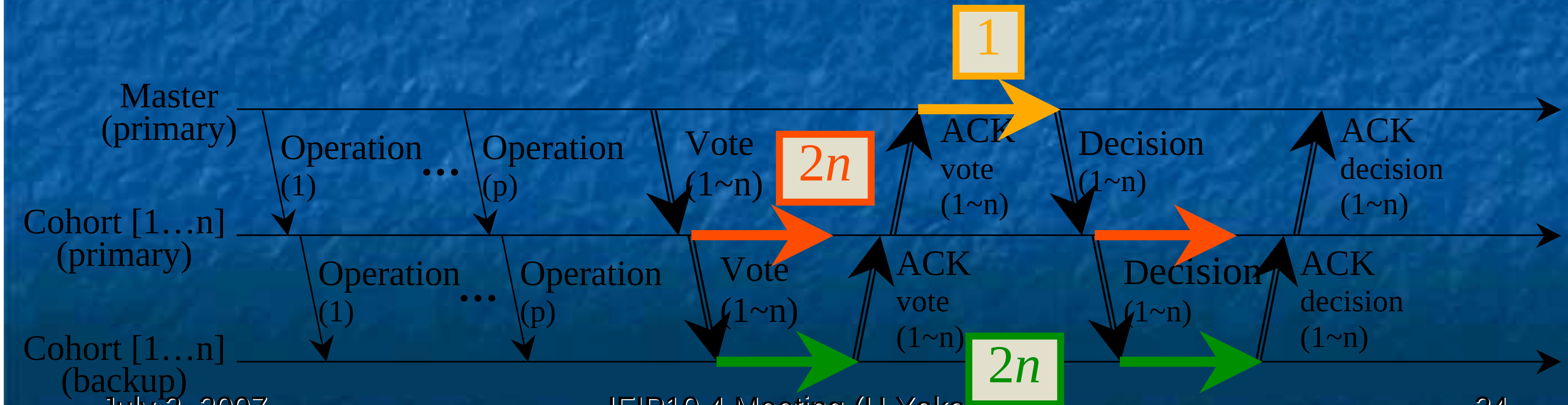
- Number of message $2p + 8n$
  - A master sends the messages to deliver the *p* operations to the cohorts
  - The master must exchange two rounds of messages with each cohort at commit phase
  - The primary cohort must forward messages to its backup cohort
  - The backup cohort reply to these messages during commit processing
- Number of forced log writes
  - The cohort must force-write two log records at commit
  - The master must force-write a log record for the decision
  - The backup cohort force-write two log records during commit processing



Overheads of 2PC

n: No. of cohorts
p: No. of operations

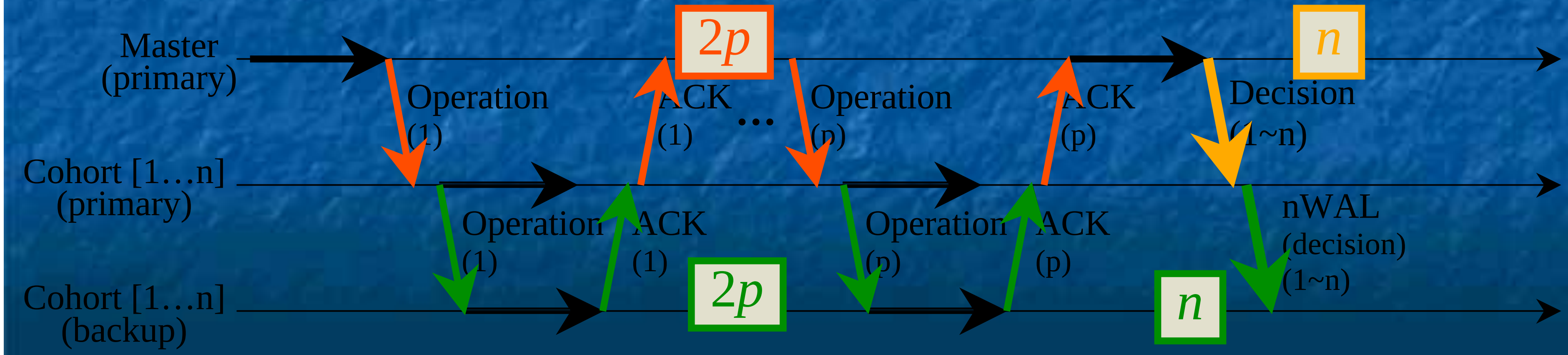
- Number of message $\boxed{2p + 8n}$
  - A master sends the messages to deliver the *p* operations to the cohorts
  - The master must exchange two rounds of messages with each cohort at commit phase
  - The primary cohort must forward messages to its backup cohort
  - The backup cohort reply to these messages during commit processing
- Number of forced log writes $\boxed{4n + 1}$
  - The cohort must force-write two log records at commit
  - The master must force-write a log record for the decision
  - The backup cohort force-write two log records during commit processing



Overheads of EP

n: No. of cohorts
p: No. of operations

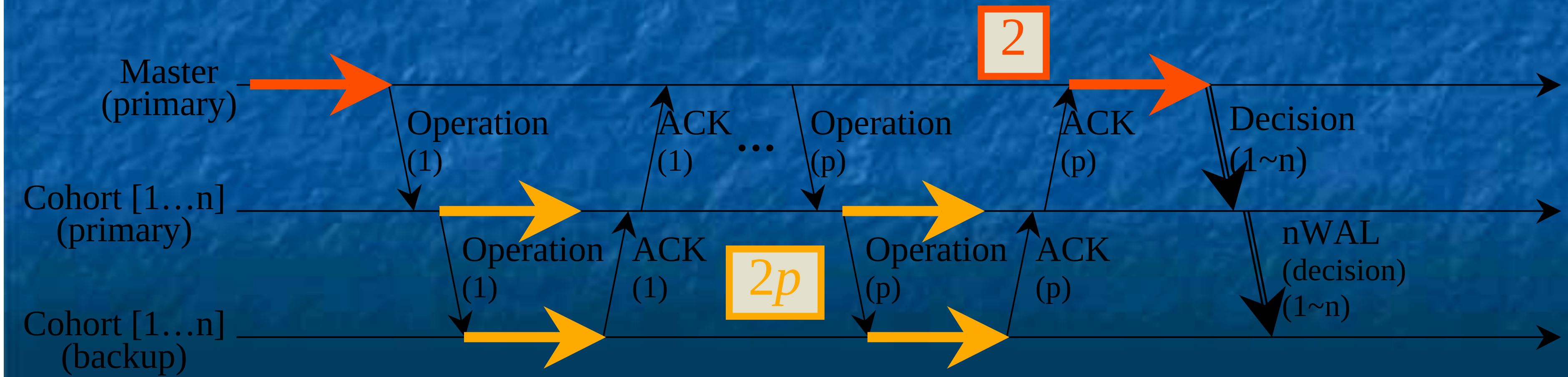
- Number of message $\boxed{4p + 2n}$
  - A cohort must acknowledge that operation with a message to a master on every operations
  - The cohort sends acknowledgement only for aborted transactions
  - The primary cohort must transmit its operations and the final decision of a transaction to the backup cohort
- Number of forced log writes
  - The master must force-write a membership log record at the beginning of a transaction and a decision log record in the commit phase
  - The cohort must force-write its log record together with its updated data to stable storage on every operation



Overheads of EP

n: No. of cohorts
p: No. of operations

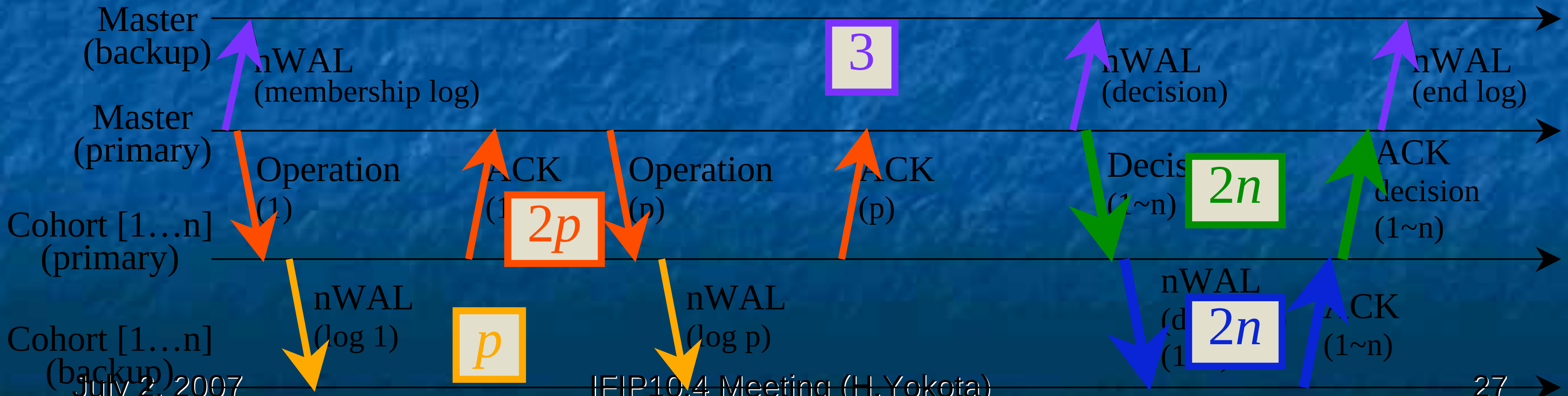
- Number of message $\boxed{4p + 2n}$
  - A cohort must acknowledge that operation with a message to a master on every operations
  - The cohort sends acknowledgement only for aborted transactions
  - The primary cohort must transmit its operations and the final decision of a transaction to the backup cohort
- Number of forced log writes $\boxed{2p + 2}$
  - The master must force-write a membership log record at the beginning of a transaction and a decision log record in the commit phase
  - The cohort must force-write its log record together with its updated data to stable storage on every operation



Overheads of BA-1.5PC

n: No. of cohorts
p: No. of operations

- Number of message $\boxed{3p + 4n + 3}$
  - delivers operations to cohorts and waits for their replies
  - the cohort issues a log using async-nWAL to its backup including the operation
  - the master broadcasts the decision to all cohorts and waits for their ACKs
  - the primary cohort writes a decision log to a backup cohort and the backup cohort returns an ACK to the primary cohort
  - the primary master must send a membership log record at the beginning of a transaction and a decision log record in the commit phase and an end log record at the end of the transaction to backup master with the async-nWAL
- Number of forced log writes $\boxed{0}$
  - requires no forced disk writes of log records with the async-nWAL



Summary of Protocol Overheads

- BA-1.5PC outperforms the other two protocol
  - It does not require any forced log writes during a transaction
- The relative ranking of 2PC and EP depends heavily on *n* and *p*
  - The operation number $p = u * n$ for Fat-Btree transactions where SMOs happen on index pages
    - *u* is the number of updates to index pages
    - usually *u* is larger than two in such an SMO
  - EP will incur a larger overhead than 2PC arising from its heavy delays for forced log writes during operation processing

Protocol	No. of message	No. of forced log writes
2PC	$2p+8n$	$4n+1$
EP	$4p+2n$	$2p+2$
BA-1.5PC	$3p+4n+3$	0

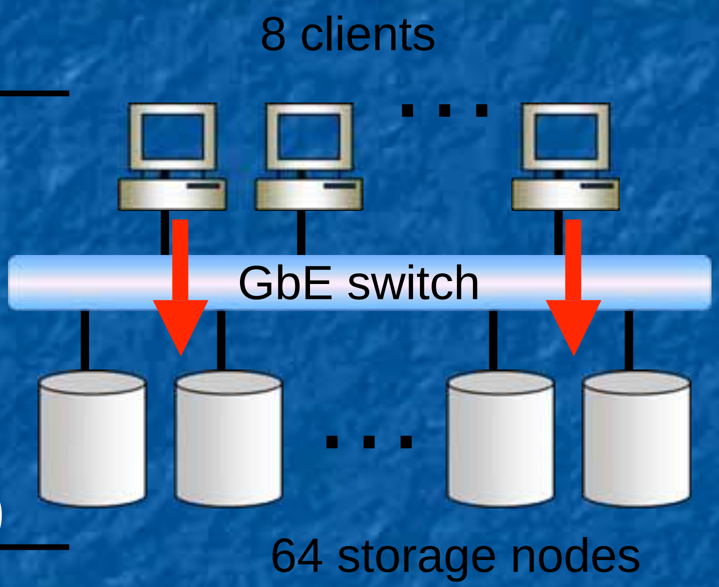
Experimentation

- The performance of proposed protocol is evaluated using an experimental system of Autonomous Disks
- Comparison targets
  - Conventional protocols: 2PC and EP
- Workloads
  - Synthetic Workloads
    - Access Requests: an access include a request to insert a key with a fixed amount of data (400 bytes)
  - Access Frequencies: uniform

Experimentation Environment

- Hardware

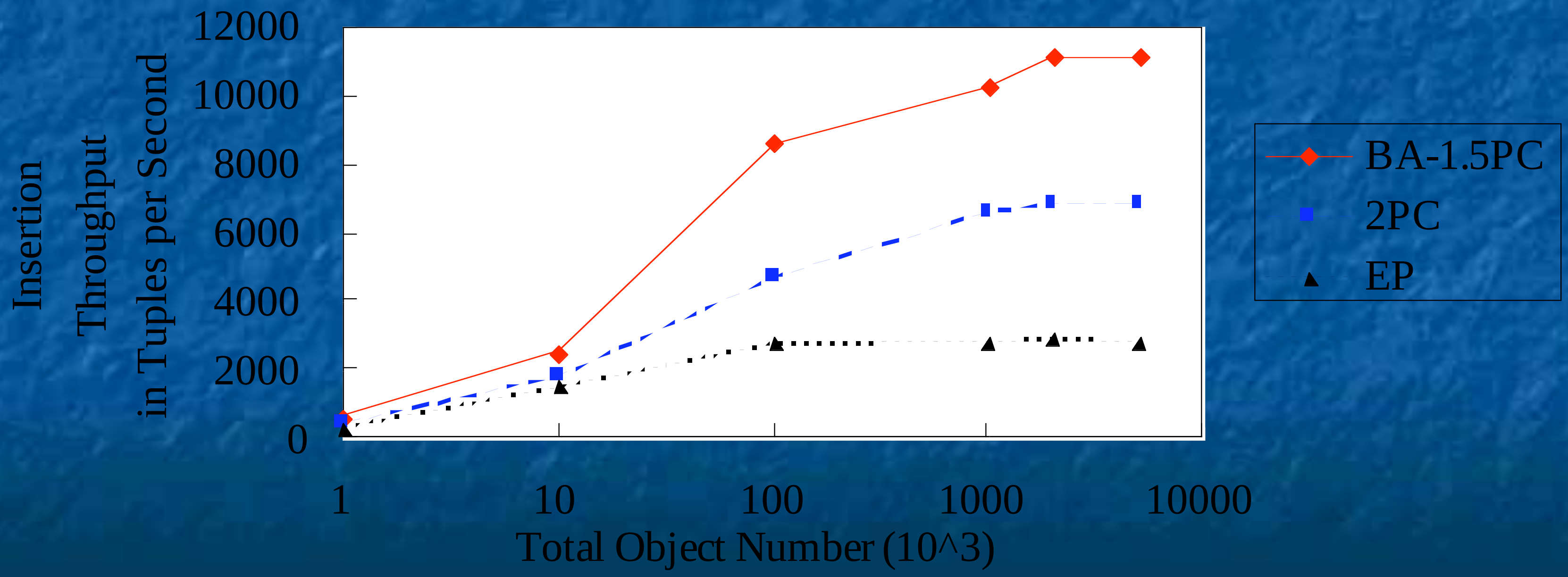
No. of PE:	**64 (Storages), 8 (Clients)**
PE:	**Sun Fire B100x**
CPU:	**Athlon XP-M1800+**
Memory:	**PC2100 DDR 1GB**
Disk:	**Toshiba MK3019GAX (30GB, 5400rpm, 2.5inch)**



- Software
  - OS: Linux 2.4.20
  - Java VM: Sun J2SE 1.4.2_04 Server VM

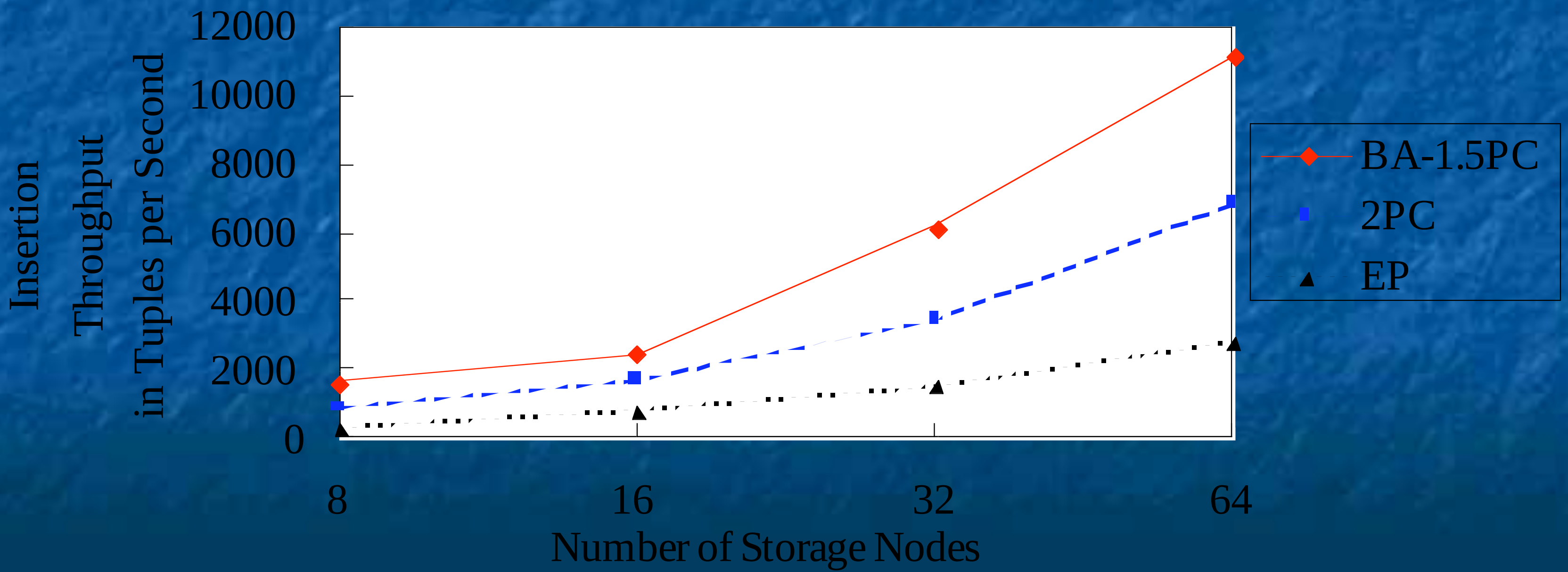
Changing Data Set Sizes (64 Nodes)

- BA-1.5PC outperforms 2PC and EP with regard to transaction throughput at all of data set sizes
  - Because it greatly reduces the message complexity during commit process, and removes the necessity to write forced disk log by the async-nWAL.



Differing Number of Storage Nodes (5M Objects)

- More storage nodes in the system allow higher insertion throughputs.
  - This is attributed to the good scalability and parallelism of the Fat-Btree index structure.
- The increase in the throughput of BA-1.5PC is larger than that of the other two protocols as the system size increases
  - BA-1.5PC is more scalable than 2PC and EP



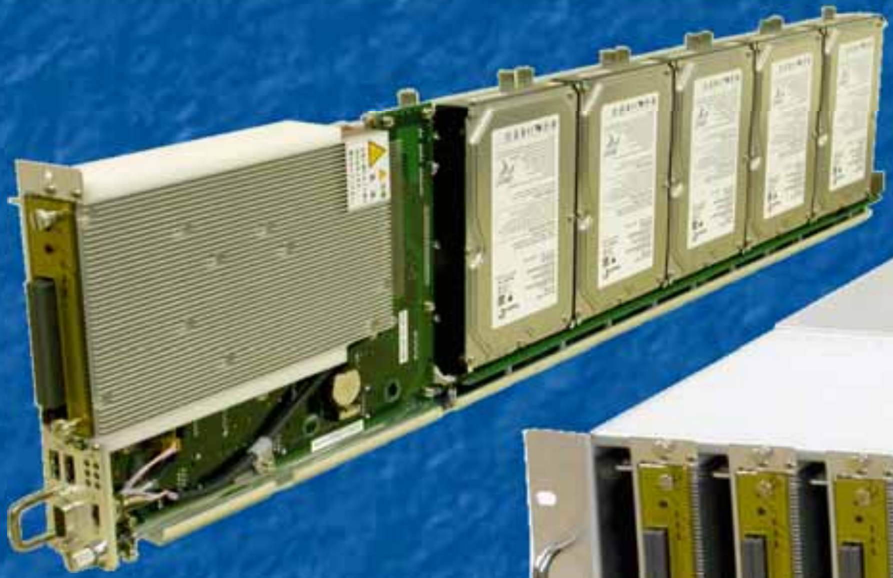
Conclusions

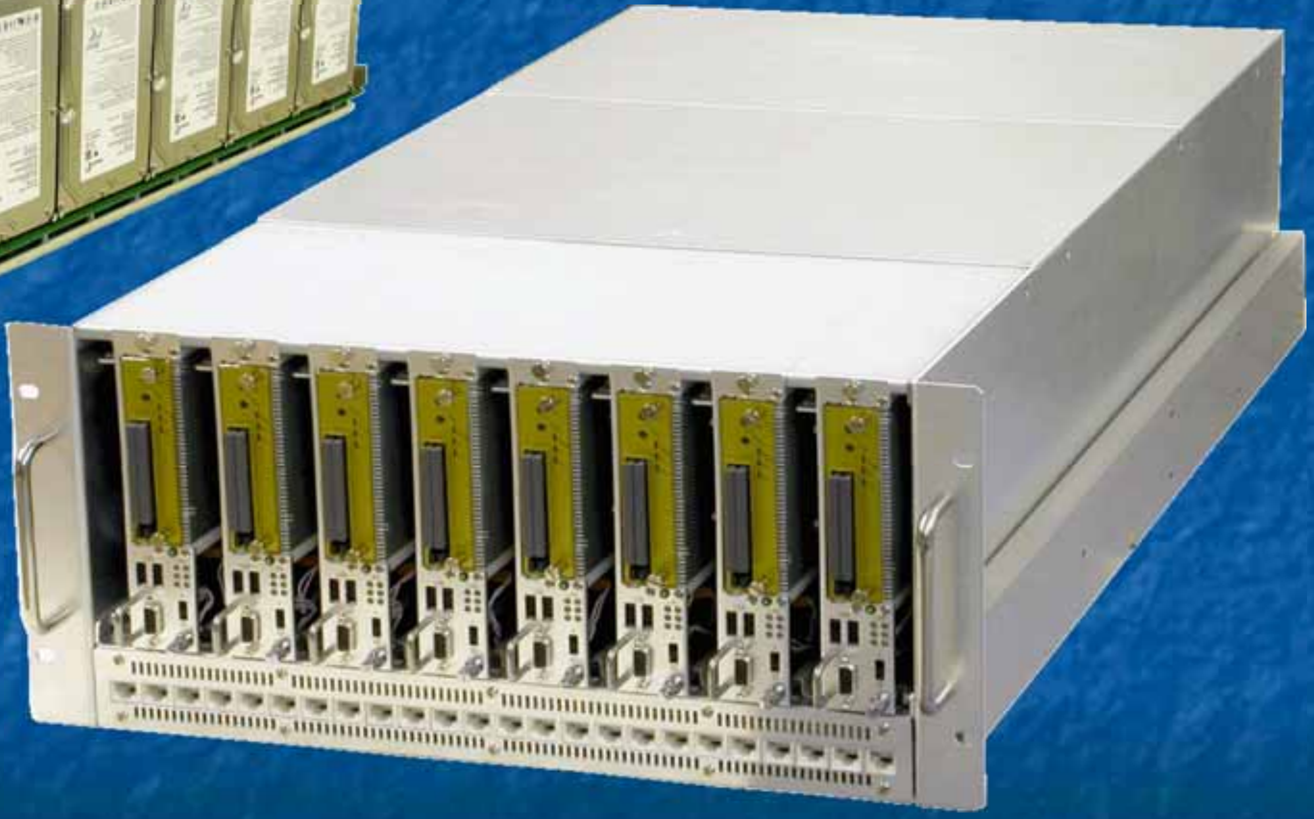
- We have proposed a new atomic commit protocol for distributed storage systems adopting the primary and backup approach to reduce the overhead of transaction processing.
- We analyzed the overhead of our protocol with the conventional protocols.
- We implemented the proposed protocol on an experimental Autonomous Disk system, and compared it with the conventional protocols.
  - It improves approximately 60% for 64 storage nodes
  - It enjoys a superior performance advantage in terms of throughput for varied dataset sizes and at different Autonomous Disk system scales.

Related Progress

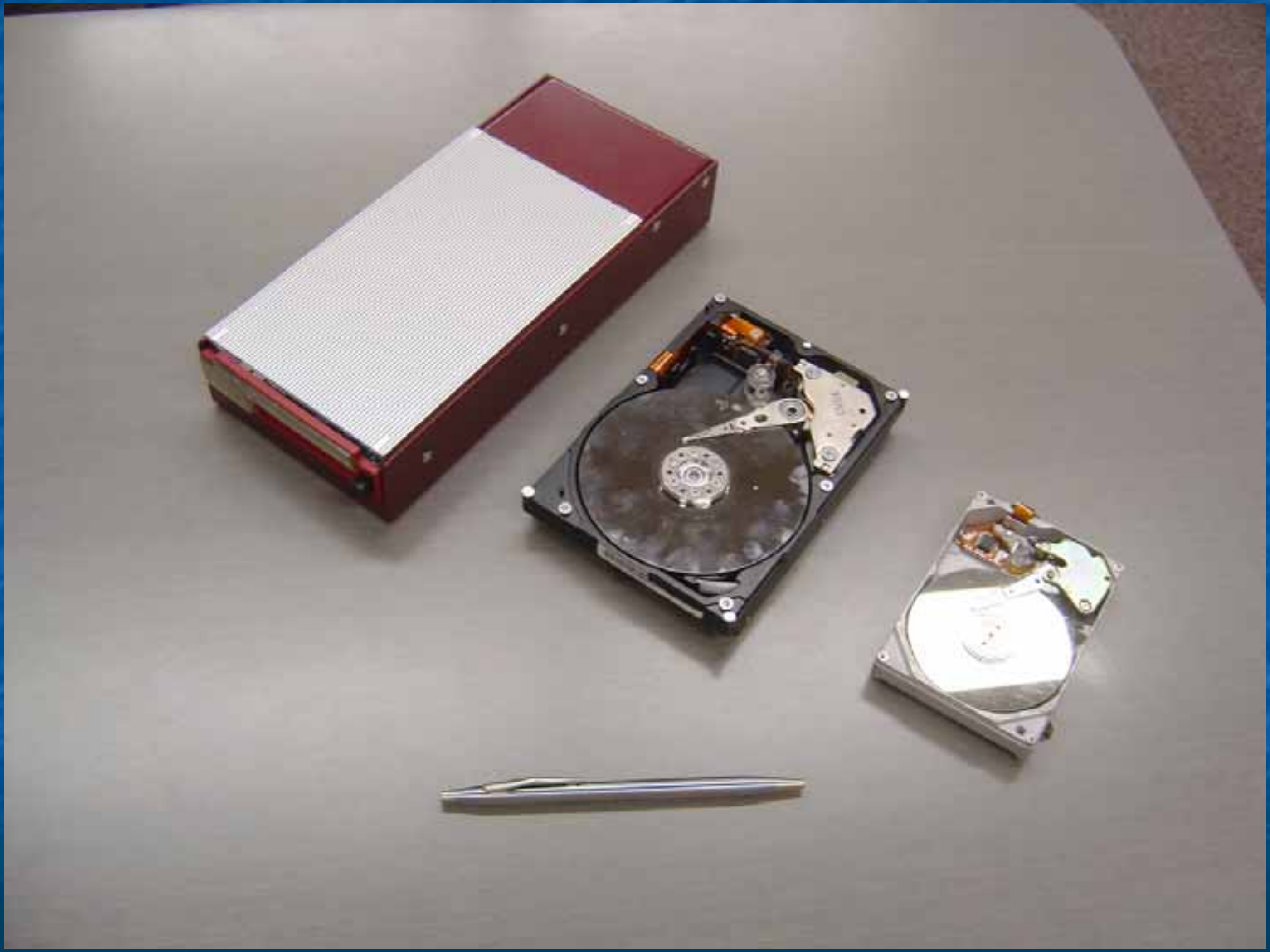
- It is interesting to exploit the processing power of the disk resident processor for advanced storage management.
- We proposed another method to balance both access load and space utilization under multi-version management control.
  - It combines the Fat-Btree with linked list structures.
- We also proposed revocation methods for encrypting on parallel disks adopting the primary backup configuration.

B (blade) type prototype





S (Small-size) type Prototype



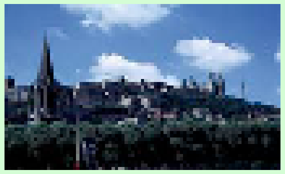
ADss 2007

International Workshop on Advanced Storage Systems 2007

in conjunction with the 2nd IEEE International Conference on Digital Information Management (ICDIM 2007)

October 28-31, Lyon, France

ICDIM



ORGANIZATION COMMITTEE

SCOPE

Storage systems are crucial for storing, maintaining, and utilizing a large amount and a wide variety of digital information such as databases, Web data, multimedia data, XML data, sensor and stream data, temporal and spatial data, etc. Next generation advanced storage systems are strongly required to handle the digital information with respect to ease of management, dependability, and security as well as high performance and scalability.

The International Workshop on Advanced Storage Systems serves as a forum for researchers and practitioners from academia, and industry, to present, discuss, and exchange their ideas that address the next generation advanced storage systems.

SUBMISSION GUIDELINES

Submitted papers should be no longer than six pages, following

Katsunori Yamaoka (Tokyo Institute of Technology)
Zhao Yuelong (South China University of Technology)

TOPICS OF INTEREST

- Advanced Disk Arrays
- Advanced File Systems
- Intelligent and High Performance Disks
- Distributed and Parallel Storage Systems
- Storage Cache
- Storage Management
- Autonomous Storage
- Dependable Storage
- Data Life Cycle Management
- Archive Systems
- Storage Networking
- Storage Benchmarks and Performance Evaluation
- Application, Security, Prototypes, and Empirical Studies of Advanced Storage Systems
- XML Data Storage and Management

SUBMISSION GUIDELINES

Submitted papers should be no longer than six pages, following the IEEE CS format found at Conference Publishing Forms, 8.5" x 11" Two-Column Format. The official language for the workshop is English, and papers can be prepared in PDF form. Manuscripts must be submitted electronically, to the following e-mail address:
adss07-submission@is.naist.jp

PROCEEDINGS

All accepted papers will be published in the workshop proceedings by the IEEE Computer Society press, and also cited in IEEE Xplore. Extended versions of the selected papers will be published in the following reviewed journals.

- Journal of Digital Information Management (JDIM) (ISSN 0972-7272)
- Journal of Information Assurance and Security (JIAS) (ISSN 1554-1010)
- International Journal of Computational Intelligence Research (ISSN 0973-1873)
- International Journal for Infonomics (IJI) (ISSN 1742-4712)
- International Journal of Internet Technology and Secured Transactions (IJITST) (ISSN 1748-569X)
- International Journal of Product Lifecycle Management (IJPLM) (ISSN: 1743-5110)

IMPORTANT DATES

Paper submission: July 6, 2007 **Notification:** August 10, 2007 **Camera-ready:** August 17, 2007

WEBSITE: http://yokota-www.cs.titech.ac.jp/crest/adss2007/ **CONTACT:** adss07@is.naist.jp

Sponsored by IEEE EMS **Supported by** JST Japan Science and Technology Agency

Thank you for your attention

The Server Blade System for Simulation