Before-image log, checkpoints, crashes

Gus Björklund. Progress.

PUG Challenge Americas, 9-12 June 2013

PROGRESS
software

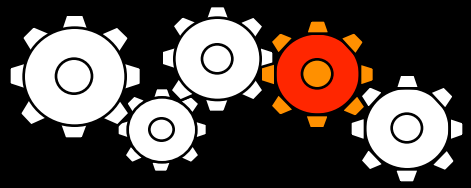
abstract

In this talk we examine the "before-image file", what it's for, how it works, and how you can configure it properly. You might get answers to questions that have been troubling people for over $25 * 10^{-2}$ centuries:
Why doesn't the before-image file have before-images ?
Why aren't the data on disk ever current ?
What are checkpoints ?
Why do we have them ?
When your system crashes (and they all do eventually) how can the RDBMS recreate all the data that were lost in the crash and restore your database to a consistent state?

The OpenEdge RDBMS is brought to you by

PROGRESS

SOFTWARE



Engine Crew

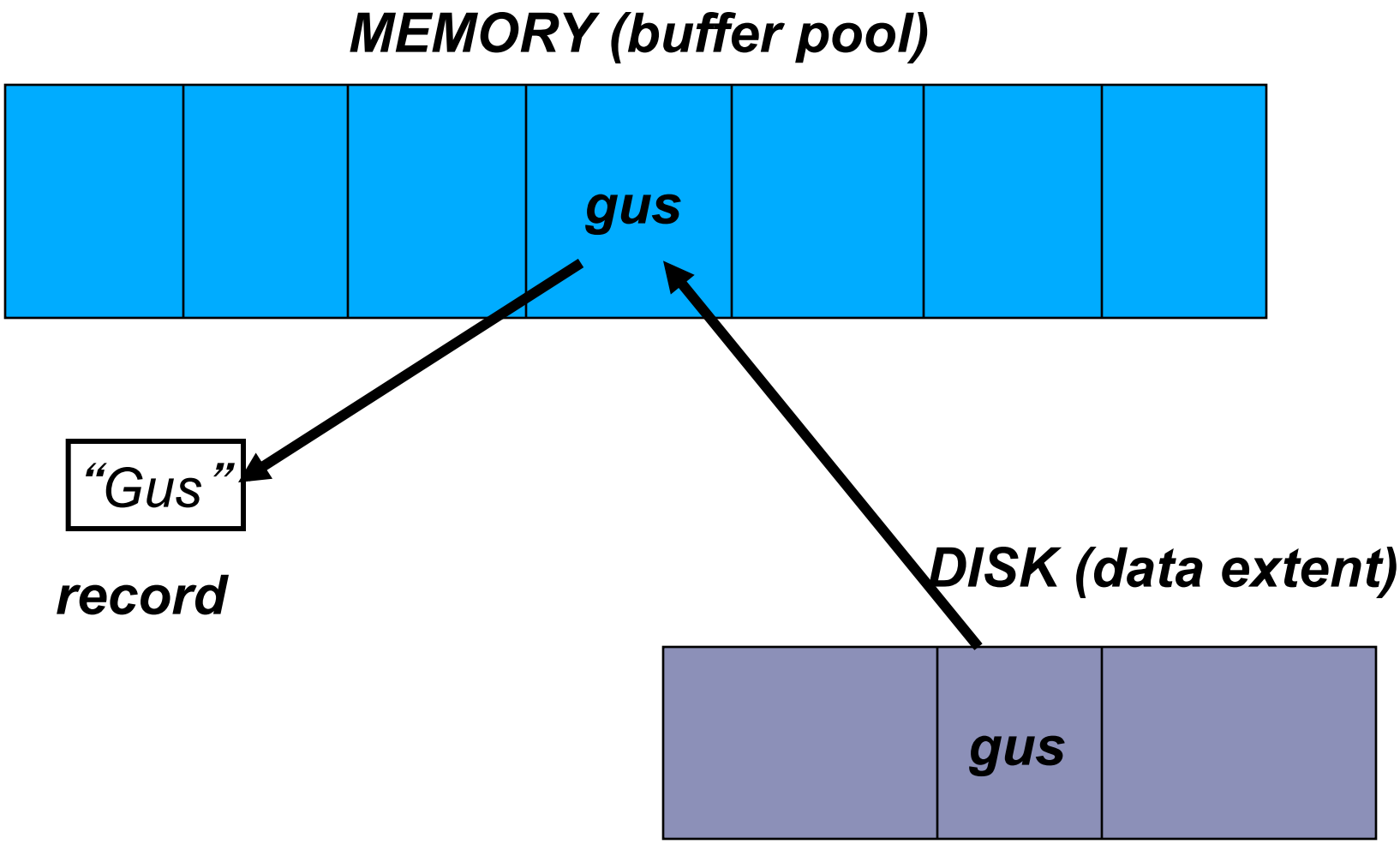
Builders of The Best RDBMS

on the Third Planet From The Sun

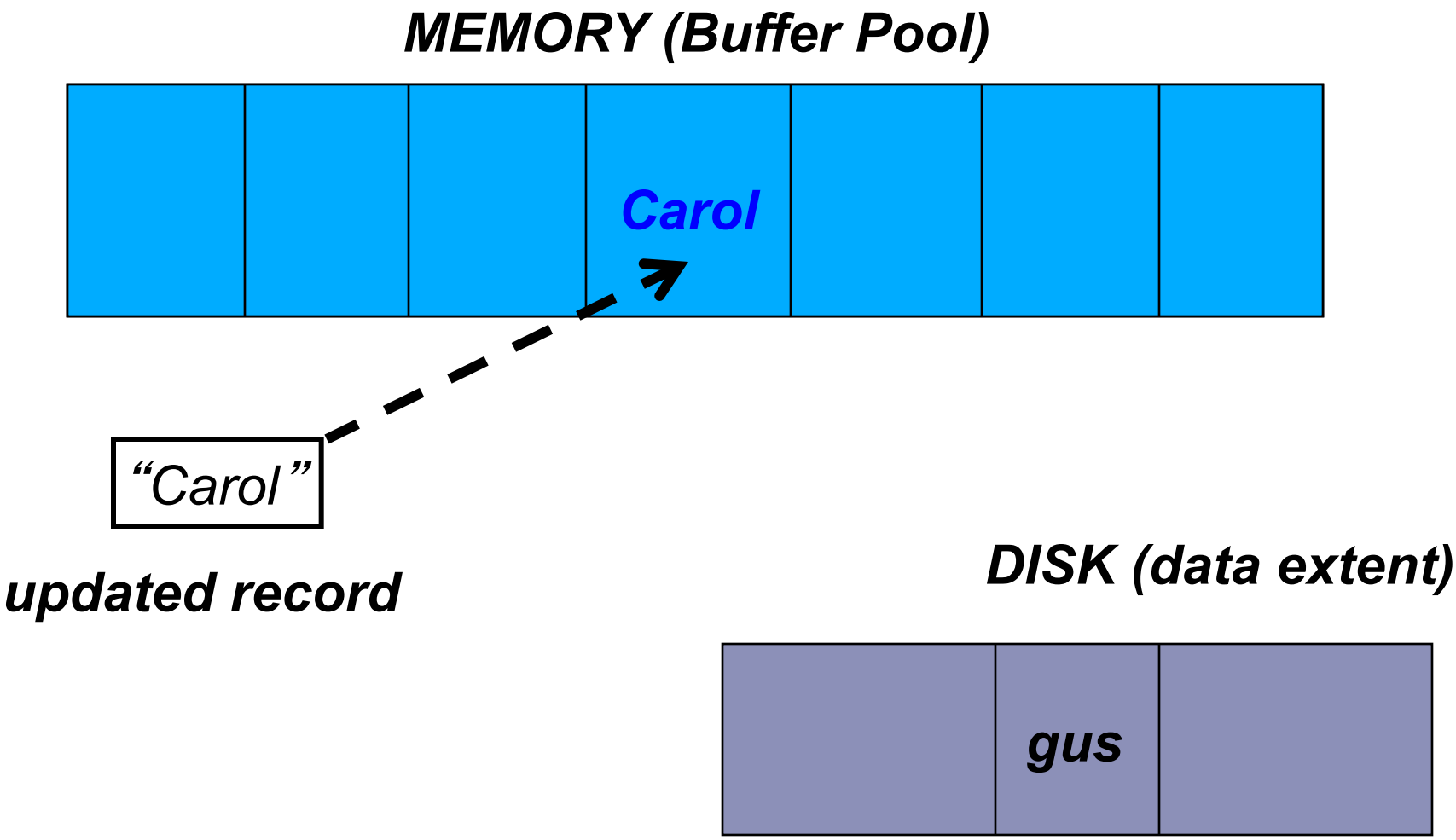
a typical data block – for records

an update

data block - before



data block - after



but….we changed memory only – not disk

PROGRESS software

- What if someone unplugs server to plug in vacuum cleaner?
- What if we want to undo (rollback) ?
- What if we make several more changes and only one block of a fragmented record chain is written to disk to make room in the buffer pool ?
- What if an asteroid wipes out all the data centers?

but….we changed memory only – no disk write

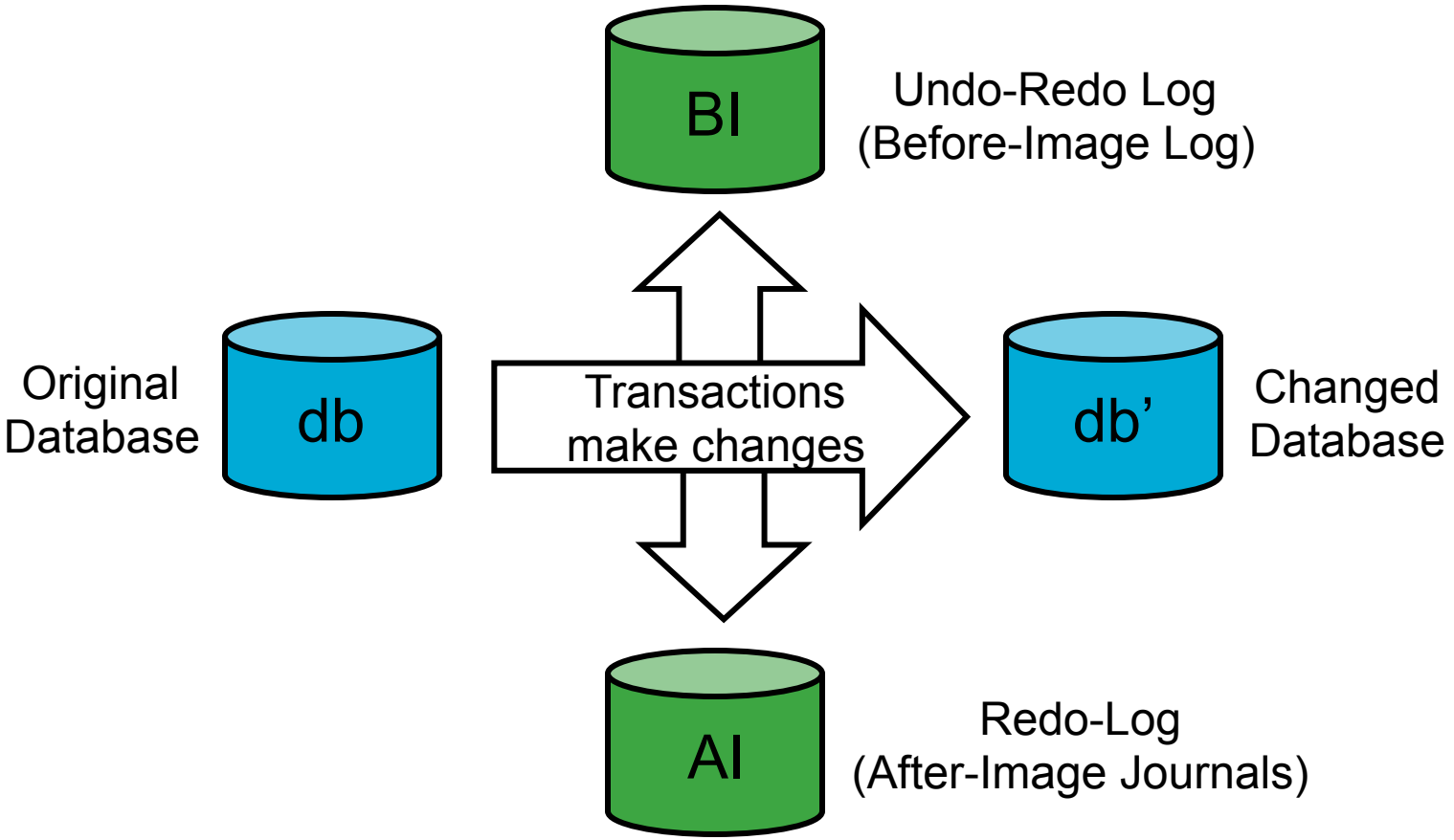
PROGRESS software

- What if someone unplugs server to plug in vacuum cleaner?
  - the change will be lost
- What if we want to undo (rollback) ?
  - we don't know the old value or how to undo
- What if we make several more changes and only one block of a fragmented record chain is written to disk to make room in the buffer pool ?
  - the database will be corrupted
- What if an asteroid wipes out all the data centers?
  - the database will disappear completely

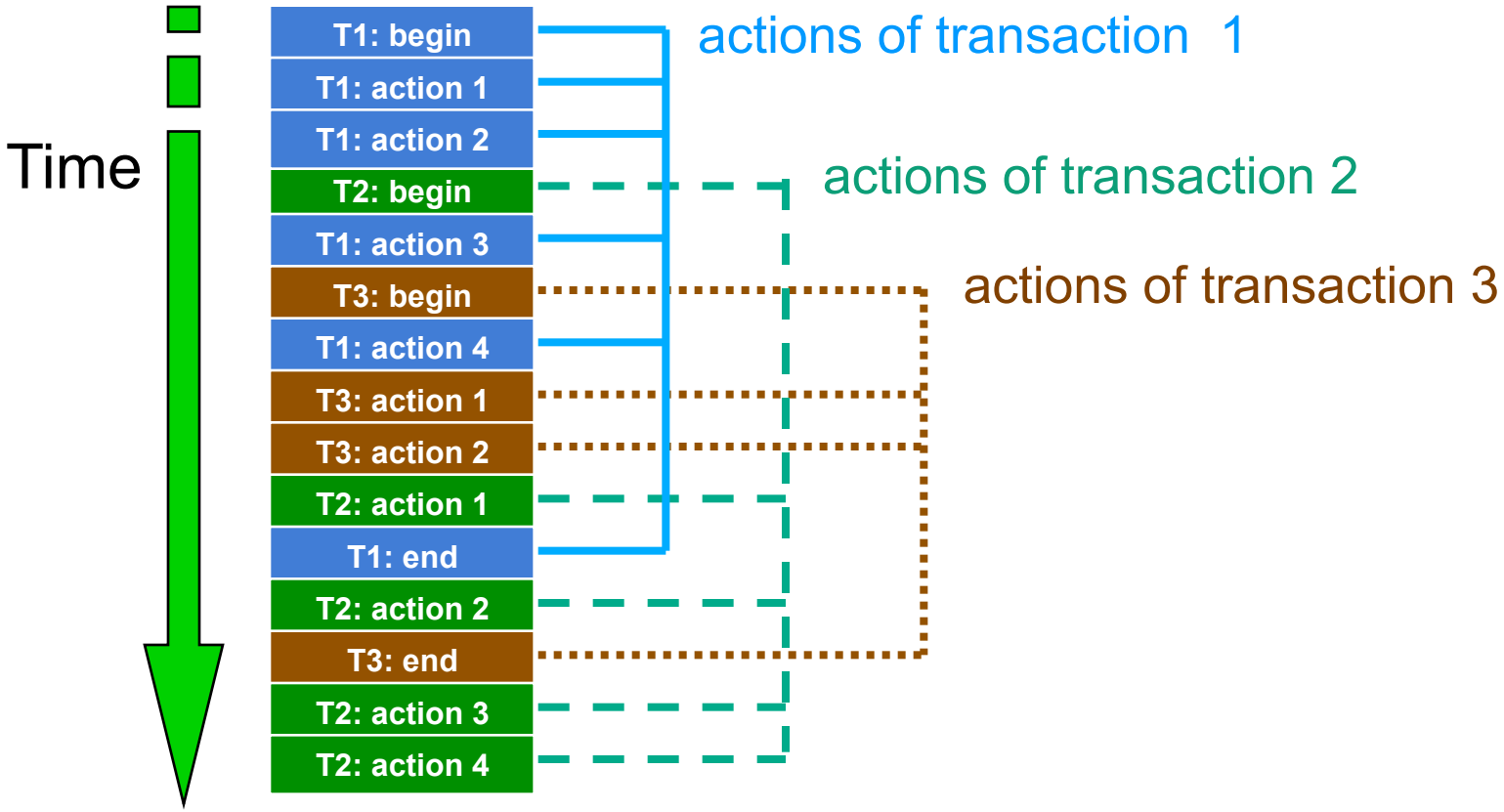
these are all bad things (tm)

transaction logging to the rescue!

Two transaction logs



transaction log records (aka “notes”)



notes form a complete history of everything

log records (notes)

PROGRESS software

- generated for every change to database.
- each describes exactly one change to one database block.
  - almost - there are log records that describe changes to purely memory-resident data structures like the transaction table
- apply only to specific version number of block
- some operations require more than one change
  - index splits, multi-block records
- written in same order changes are executed.
- notes from concurrent transactions are mixed together.

Undo-Redo (BI) Log Records

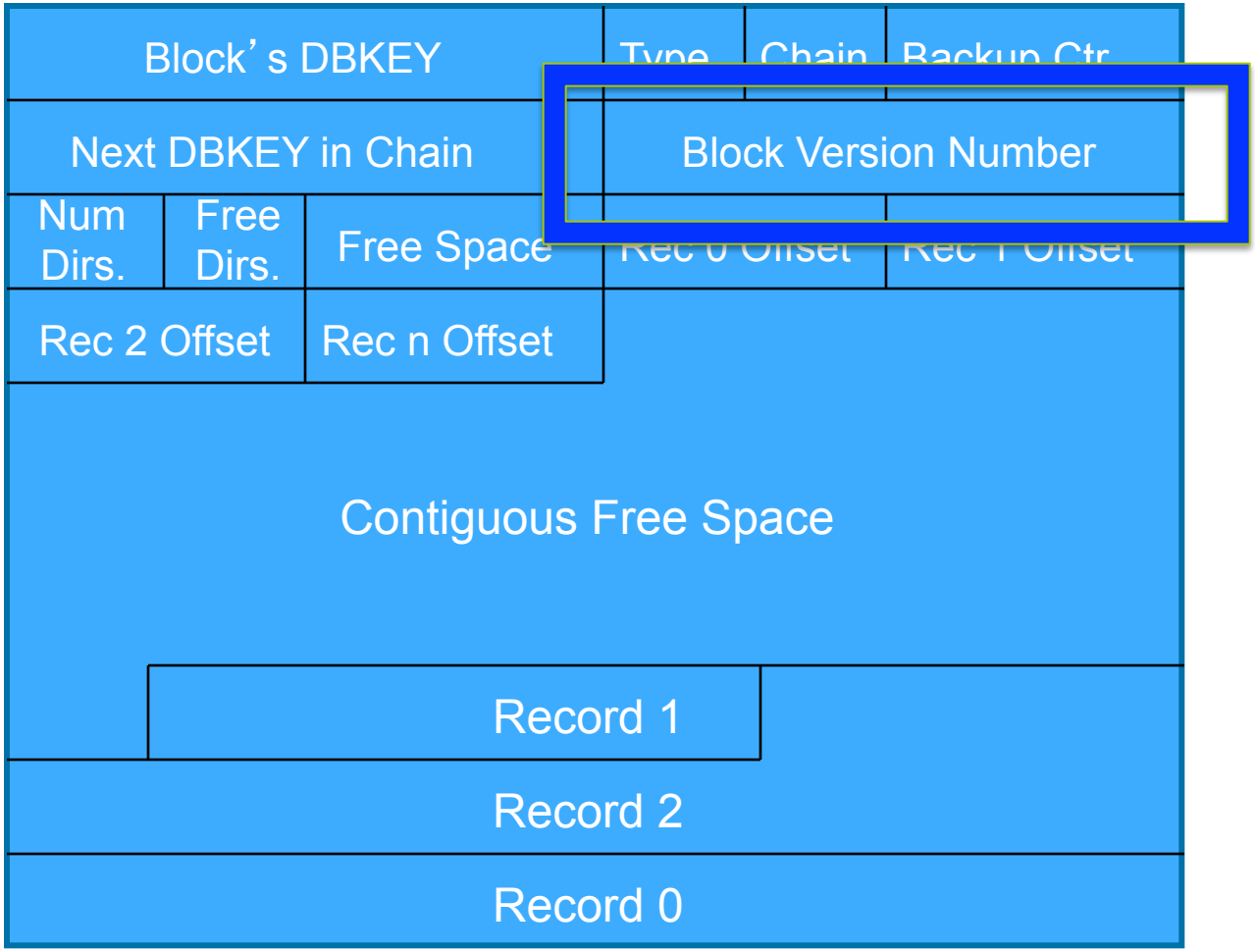
PROGRESS software

- each log record (or “note”) contains:
  - area number
  - database block number (its dbkey)
  - database block *version number*
  - note type - identifies operation to do
  - any information needed to undo the operation
    - in case we have to roll back
  - any information needed to redo the operation
    - in case we lose what is in memory

an update, with notes

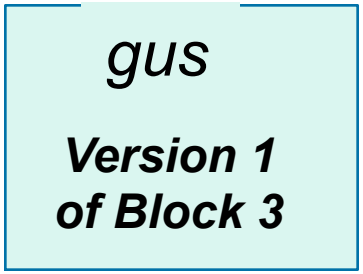
a typical data block – for us to update

a typical data block – for us to update

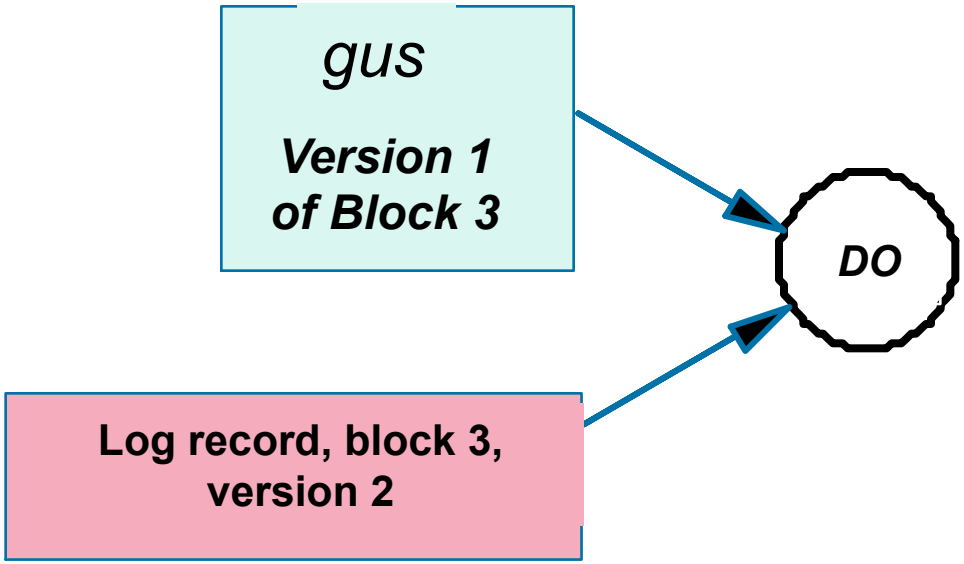


updating a block - revisited

PROGRESS software

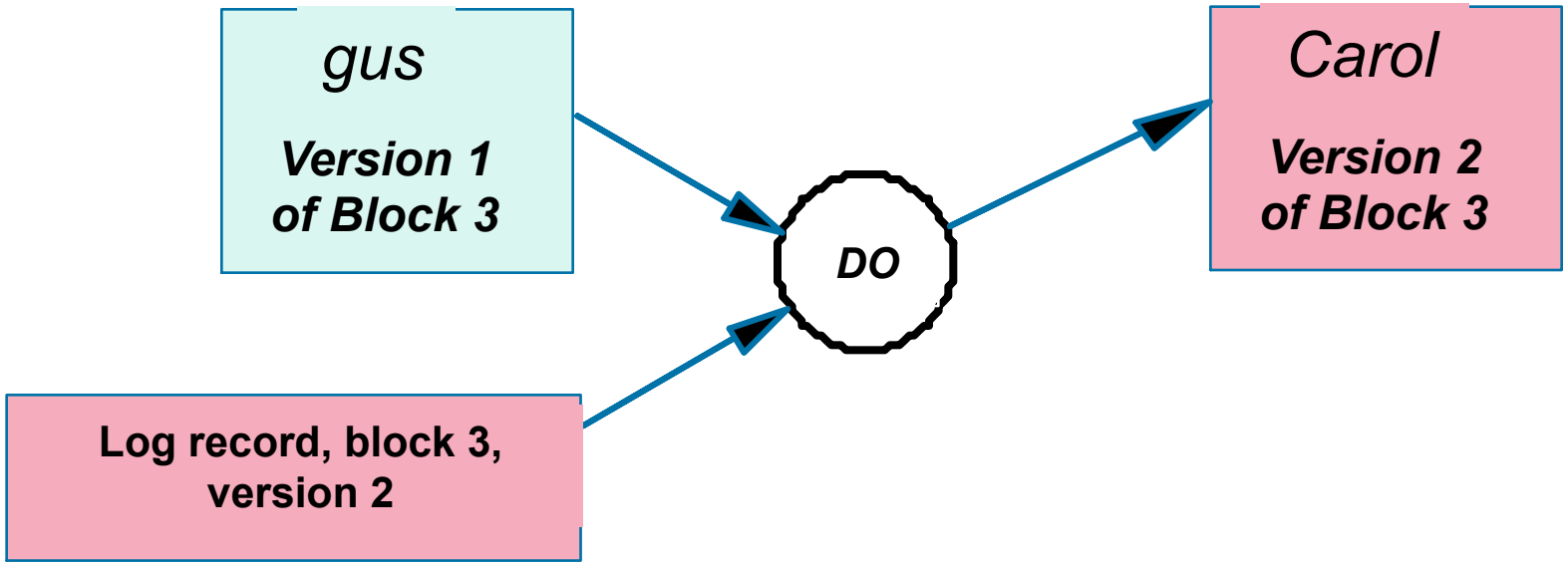


updating a block - revisited



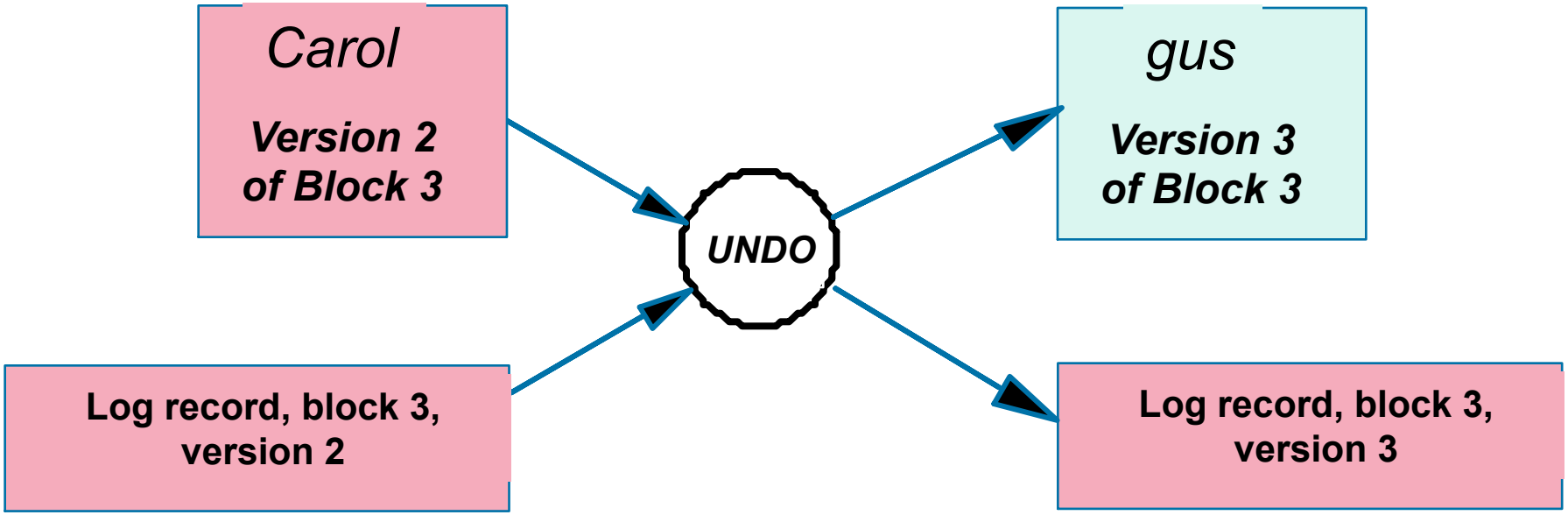
updating a block - revisited

new data values
new version



updating a block - undoing

original data values
new version

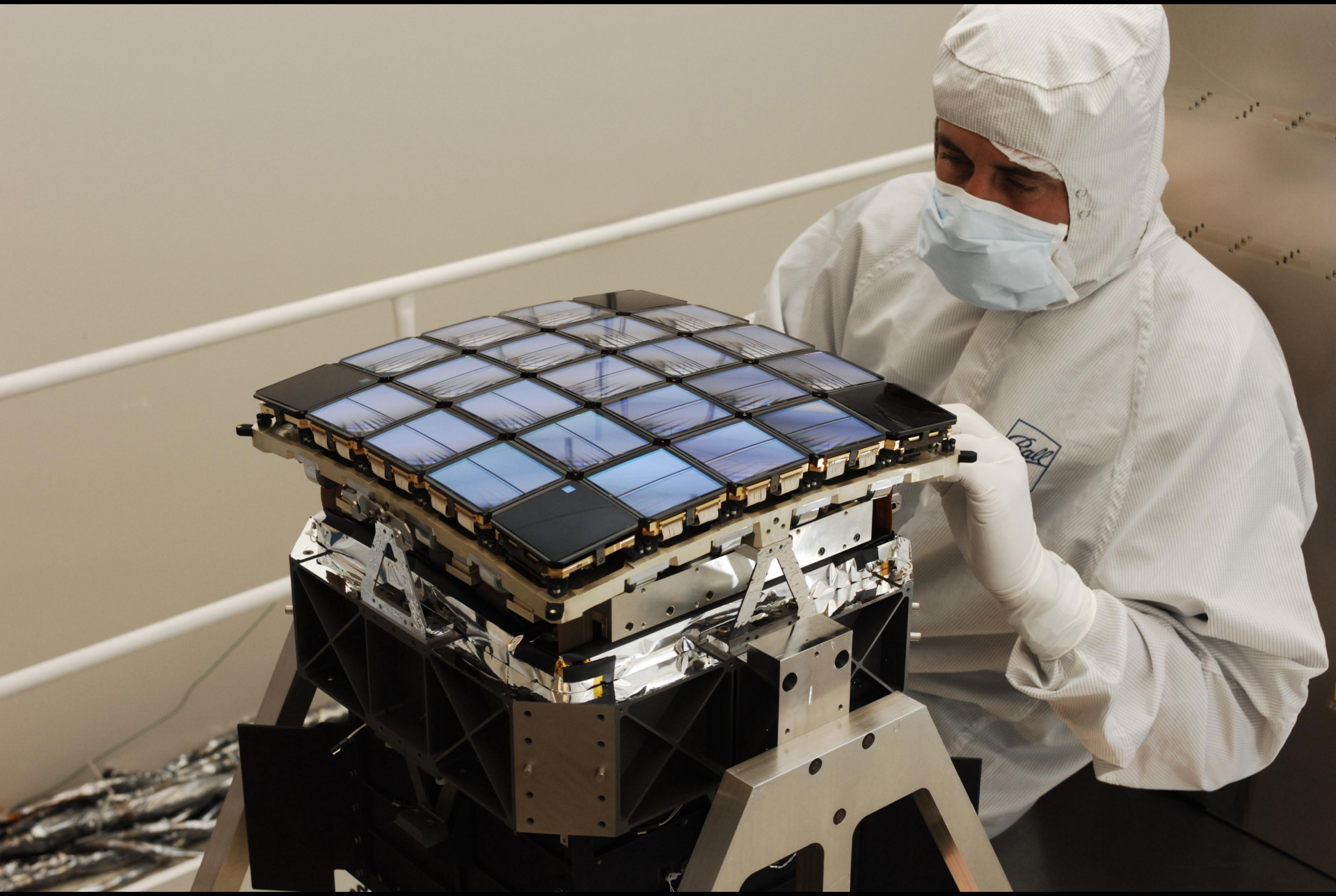


putting things back the way they were before you touched them

houston, we have a problem!

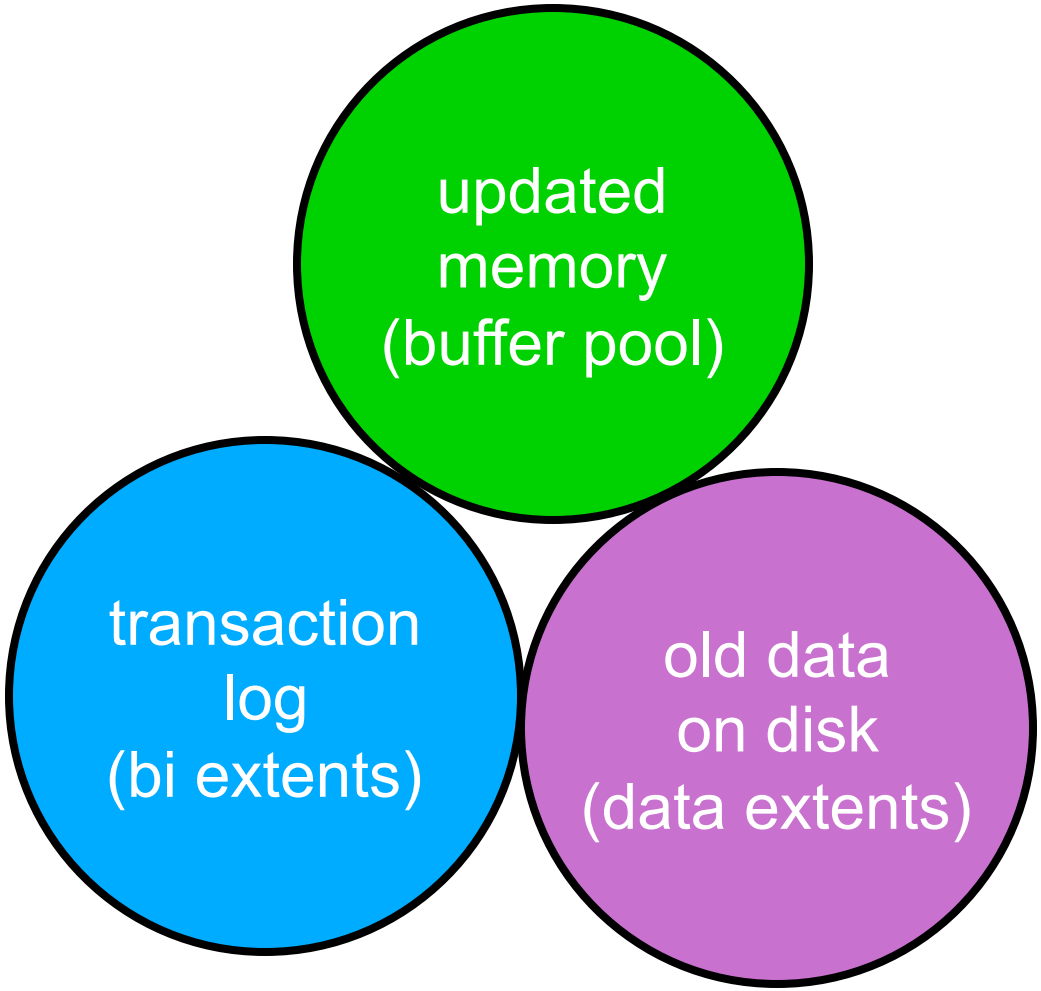
PROGRESS software

- notice that we did the change just in memory
- we are logging the changes, and we can undo if necessary, but
  - how about writing changes to disk ?
  - when ?
  - what if server unplugged ?



The checkpoint process

complete database state – in 3 part harmony



Database Checkpoints

PROGRESS software

- We have memory resident database state (updates are done in memory)
- Must update disk resident data once in a while
- Definition:
  A checkpoint is a process for making what is on disk consistent with the changed or updated database parts that are present only in memory.

  It is a process, not an event.

Benefits of Checkpointing (1)

PROGRESS software

- Smaller undo-redo (BI) transaction logs
  - Space can be re-used when the recovery information is no longer needed
- Example:
  - 1,000,000 transactions
  - 350 bytes logged per transaction

Benefits of Checkpointing (1)

PROGRESS software

- Smaller undo-redo (BI) transaction logs
  - space can be re-used when the recovery information is no longer needed
- Example:
  - 1,000,000 transactions
  - 350 bytes logged per transaction
  - so:
    - about 350 megabytes of log data
    - can execute thousand times more transactions a day
    - How much space will that take?
    - Most transactions are larger

Benefits of Checkpointing (1)

PROGRESS software

- Smaller undo-redo (BI) transaction logs
  - space can be re-used when the recovery information is no longer needed
- Example:
  - 1,000,000 transactions
  - 350 bytes logged per transaction
  - so:
    - about 350 megabytes of log data
    - could execute a thousand times more transactions a day
    - How much space will that take? → 350 gigabytes
    - Most transactions are larger

Benefits of Checkpointing (2)

PROGRESS software

- Shorter Recovery time
  - fewer changes must be repeated when a crash occurs
- Example:
  - 1,000,000 transactions
  - 3.2 disk io's per transaction
  - assume disks do about 100 io's per second
  - Arrival rate of seconds is fixed at 86,400 per day
  - So:

Benefits of Checkpointing (2)

PROGRESS software

- Shorter Recovery time
  - few changes must be repeated when a crash occurs
- Example:
  - 1,000,000 transactions
  - 3.2 disk i/o's per transaction
  - modern disks do 100 io's per second
  - Arrival rate of seconds is fixed at 86,400 per day
  - So:
    - 320,000 seconds (3.7 days) to recover
    - What if you had to recover a thousand times more?

Drawbacks of Checkpointing

PROGRESS software

- Not free !
  - Requires (some) extra processing
  - Requires (some) extra io
  - Takes (some) time
  - Can freeze all database updates for a (short) time

well worth the costs !

Checkpoint Process

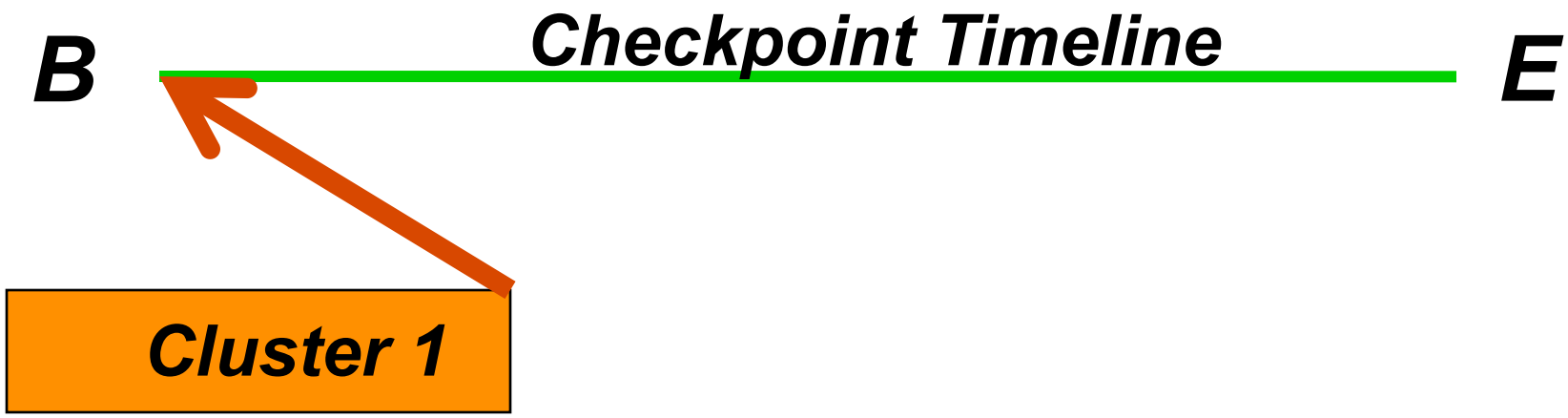
- There are 3 phases to a checkpoint

Checkpoint Process

- There are 3 phases to a checkpoint
  - Beginning
  - Middle
  - and End

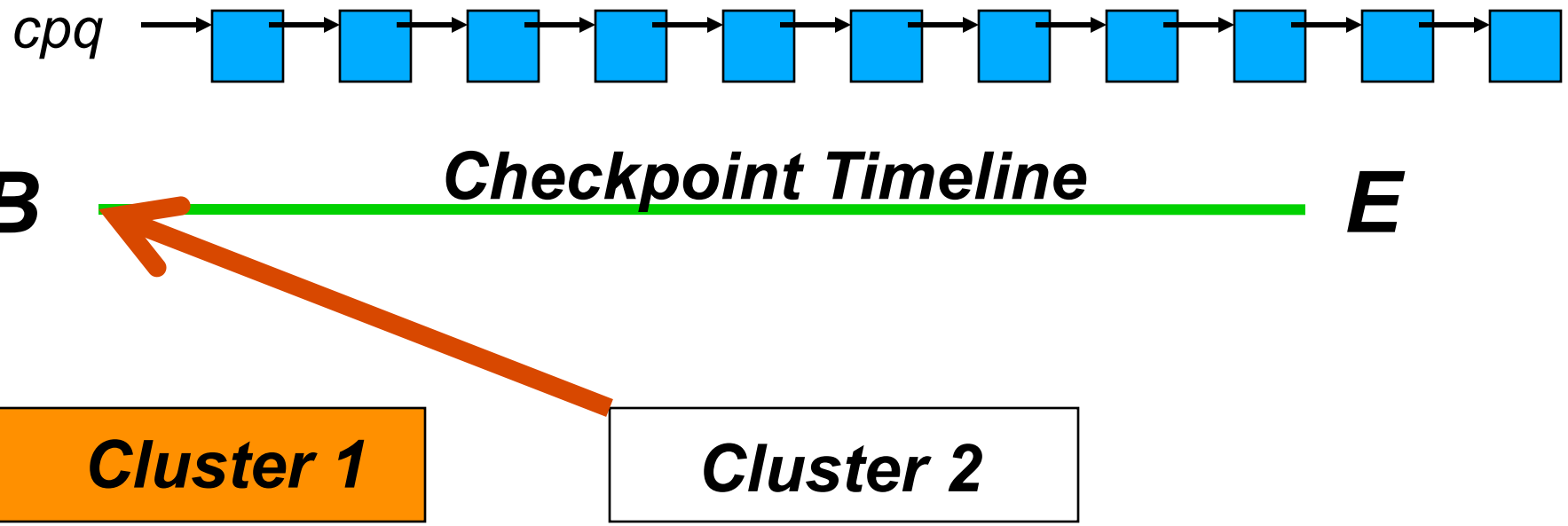
Checkpoint Phase 1 (Begin)

- Unwritten BI and AI buffers forced to disk
- All dirty blocks placed on checkpoint queue
- Next BI cluster opened
  - (may require formatting if new)



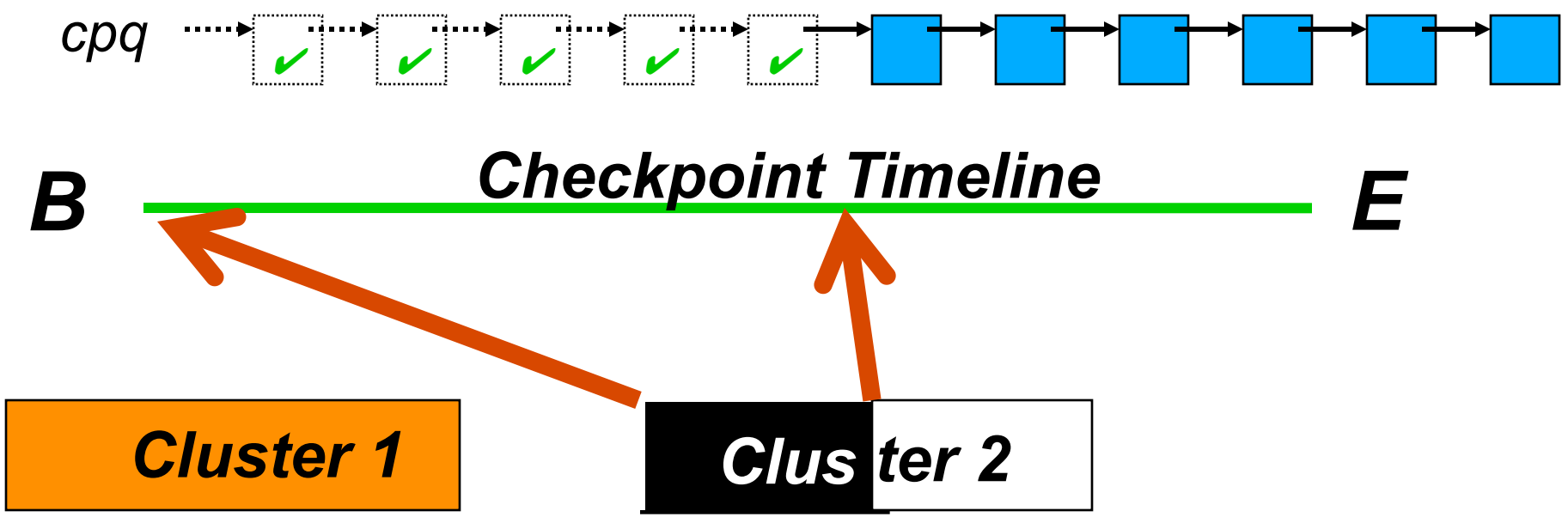
Checkpoint Phase 1 (Begin)

- Unwritten BI and AI buffers forced to disk
- All dirty blocks placed on checkpoint queue
- Next BI cluster opened



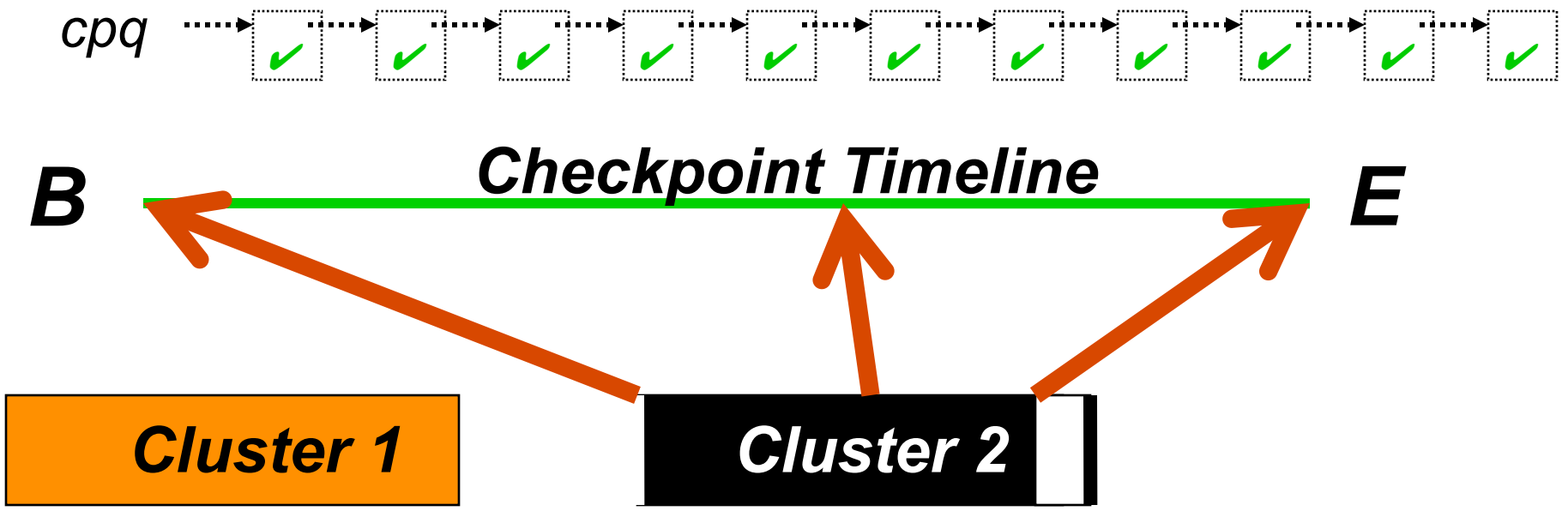
Checkpoint Phase 2 (Middle)

- Asynchronous Page Writers take blocks off the Checkpoint Queue and write them to disk.
- APW's pace themselves



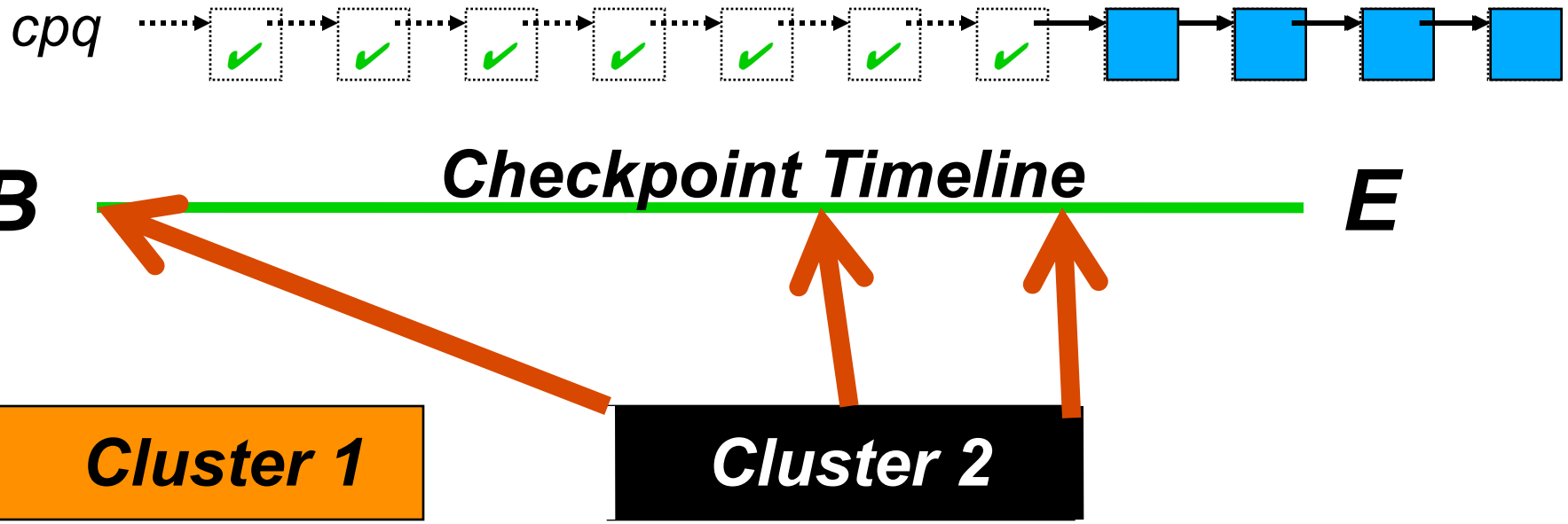
Checkpoint Phase 3 (End)

- As cluster approaches full, all blocks from checkpoint queue have been written to disk
- Checkpoint queue now empty



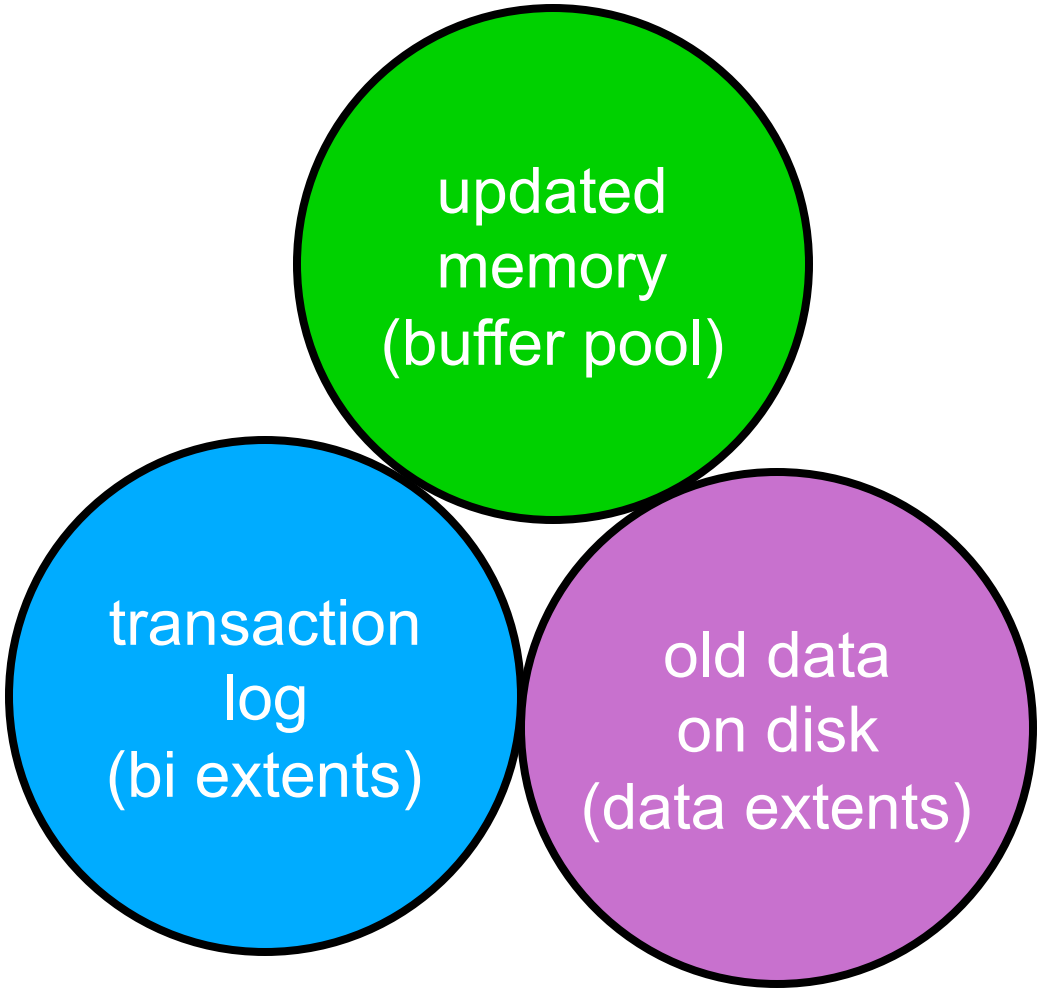
Checkpoint Phase 3 (alternate ending)

- Cluster might fill *before* queue emptied
- Now we have to flush remaining blocks
- Delay ! AND: fdatasync() calls take more time than normal – more delay

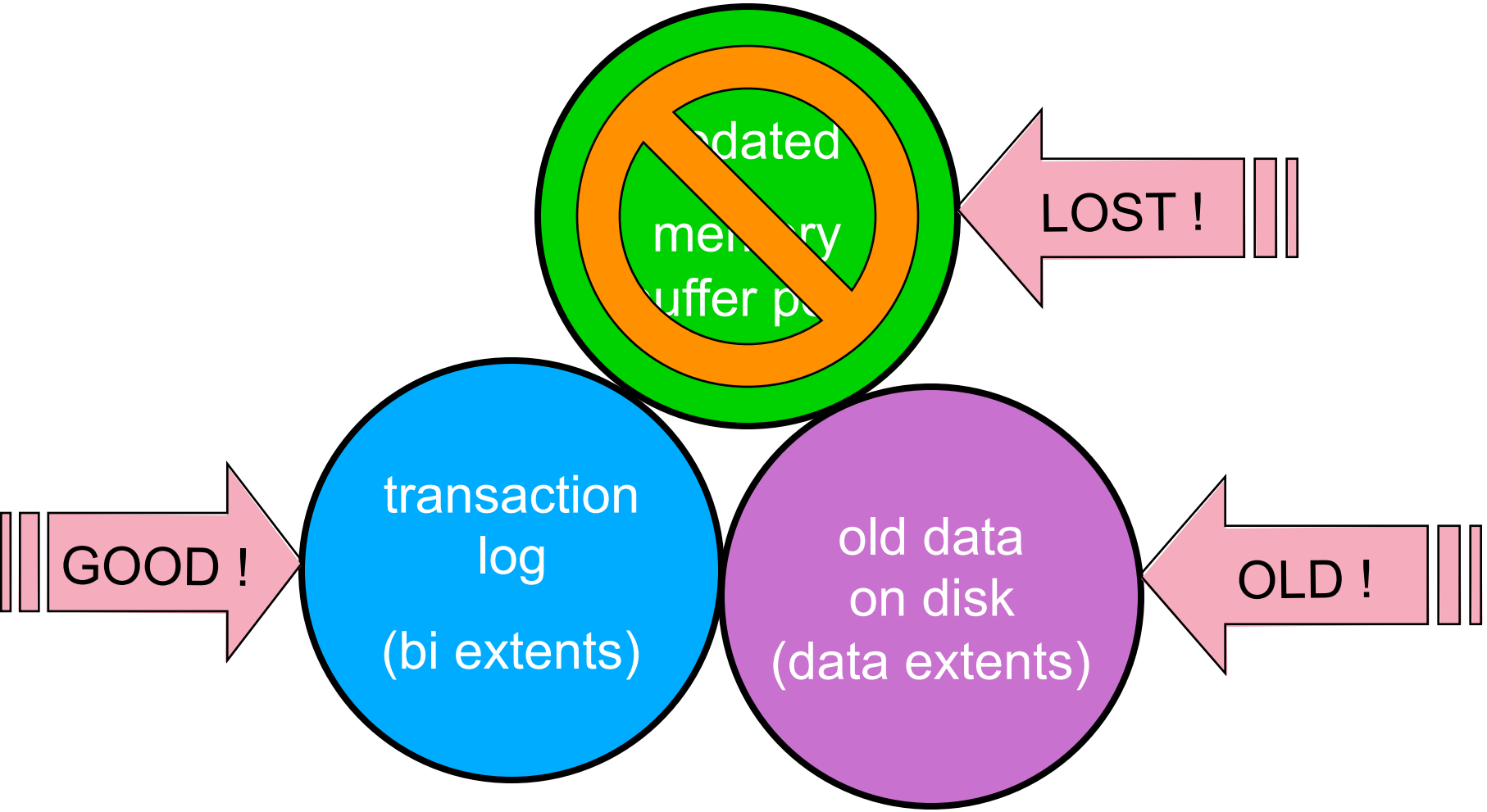


Crash recovery

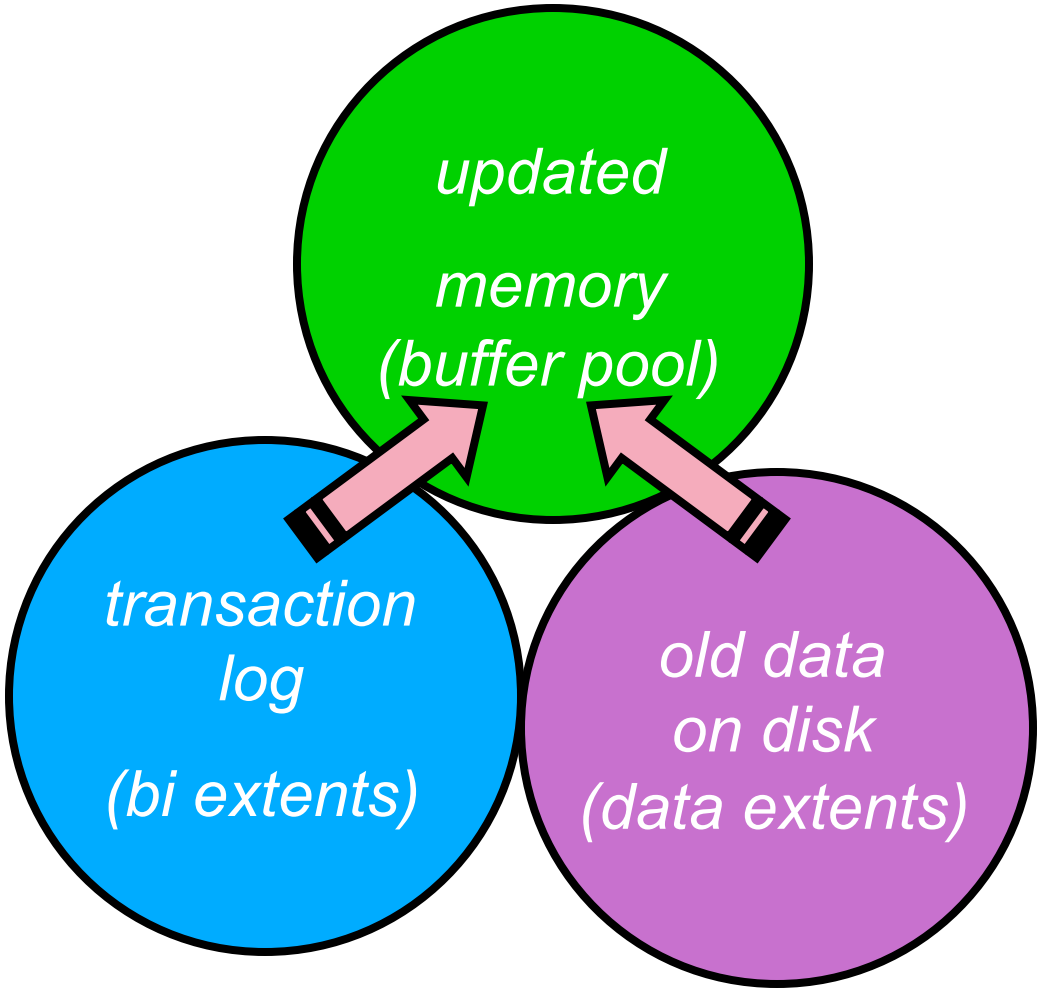
complete database state – in 3 part harmony



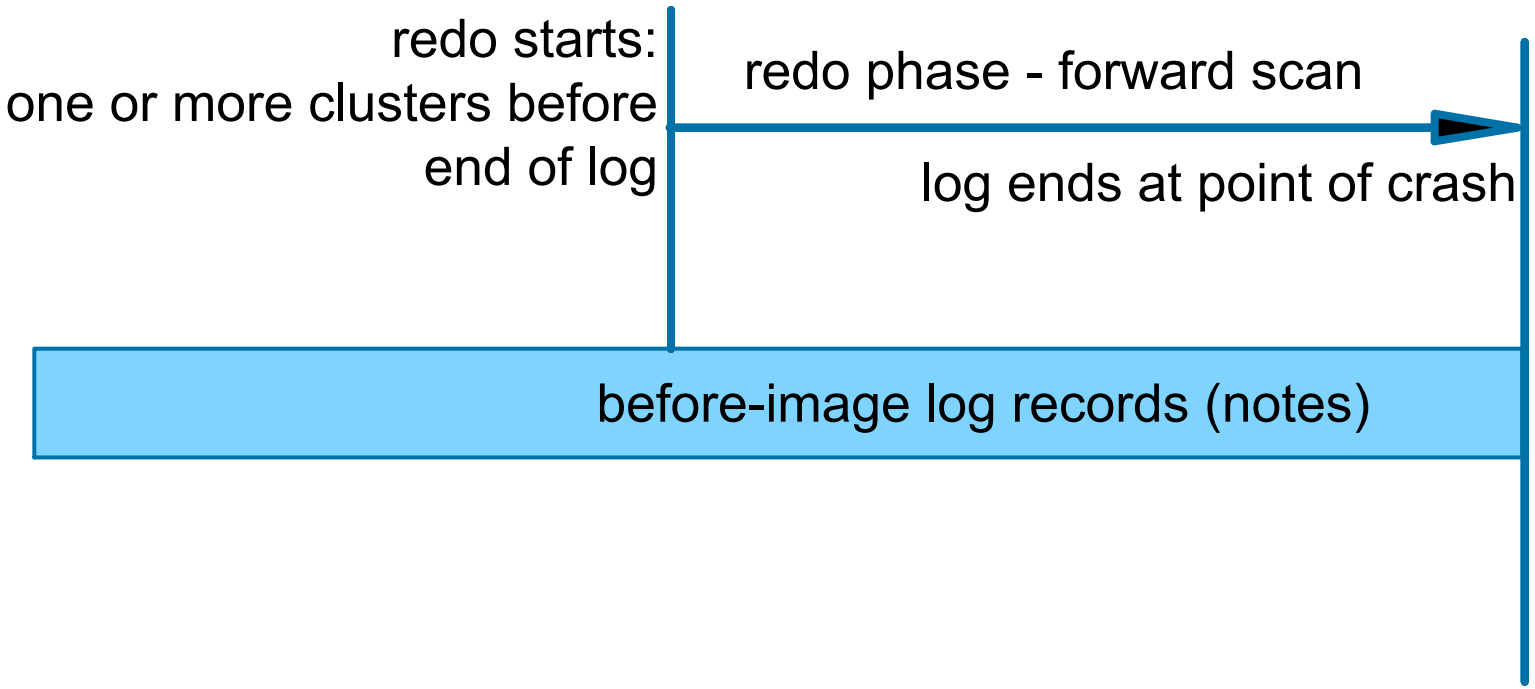
disaster strikes



reconstructive surgery

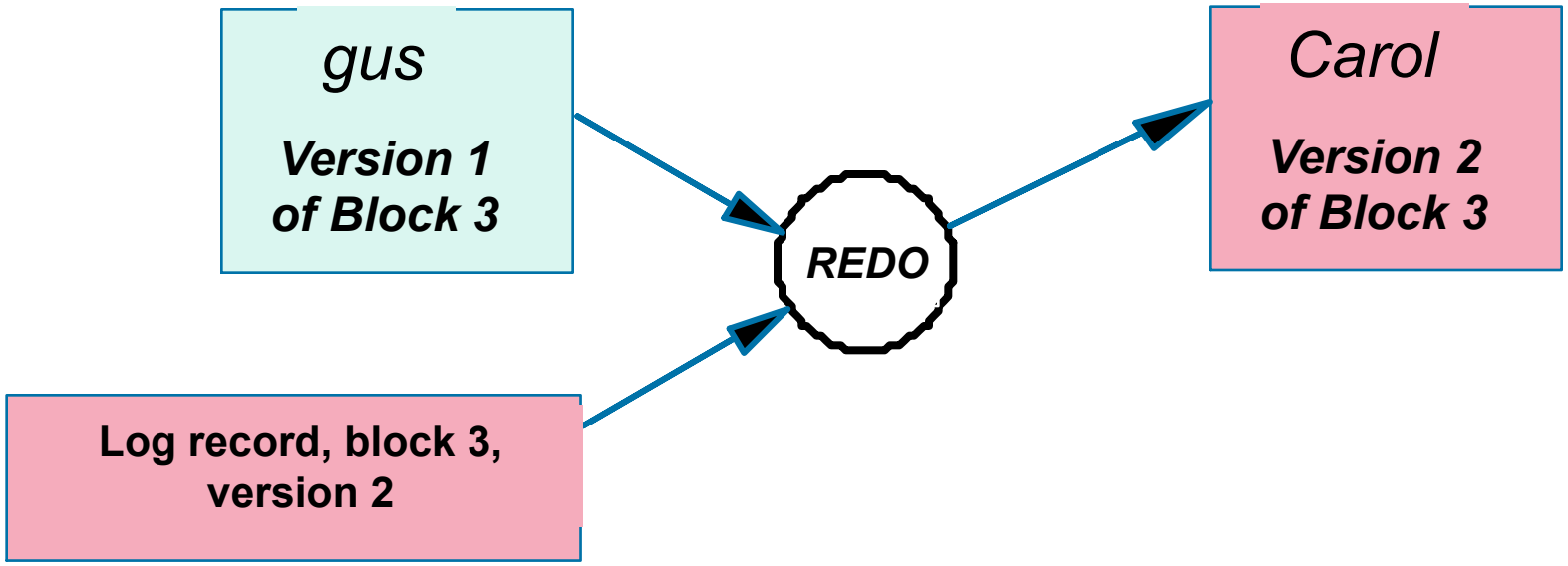


crash recovery processing

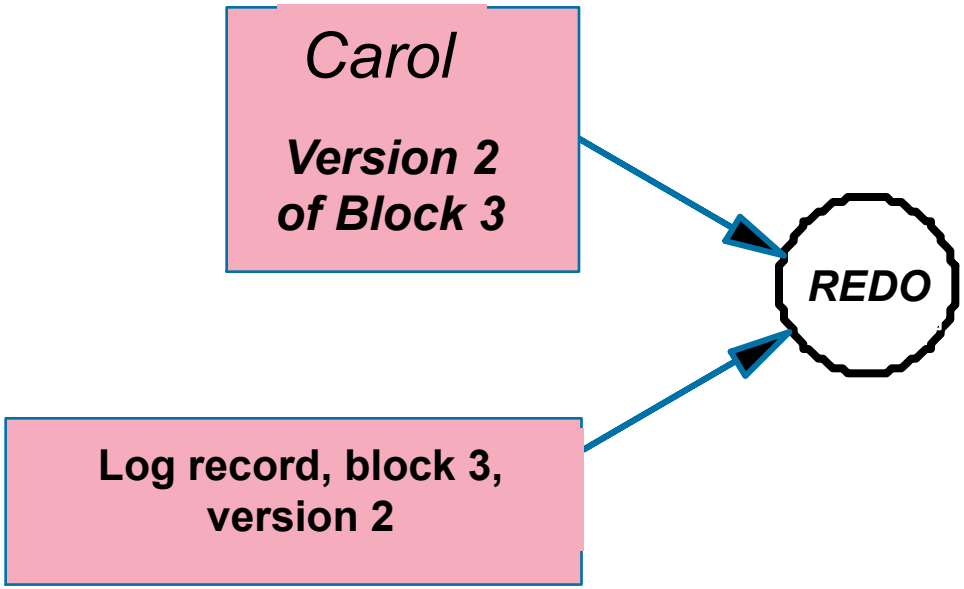


redo a change

new data values
new version



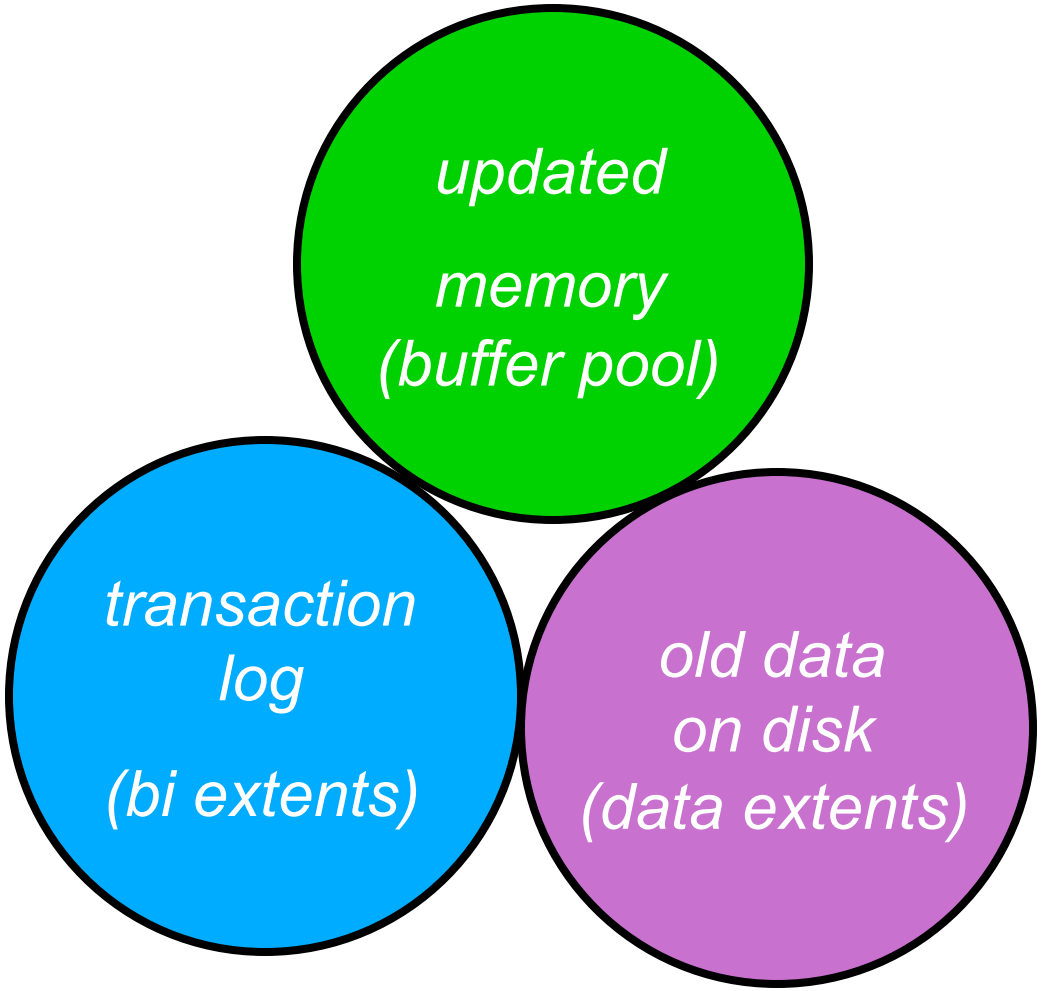
not redoing a change



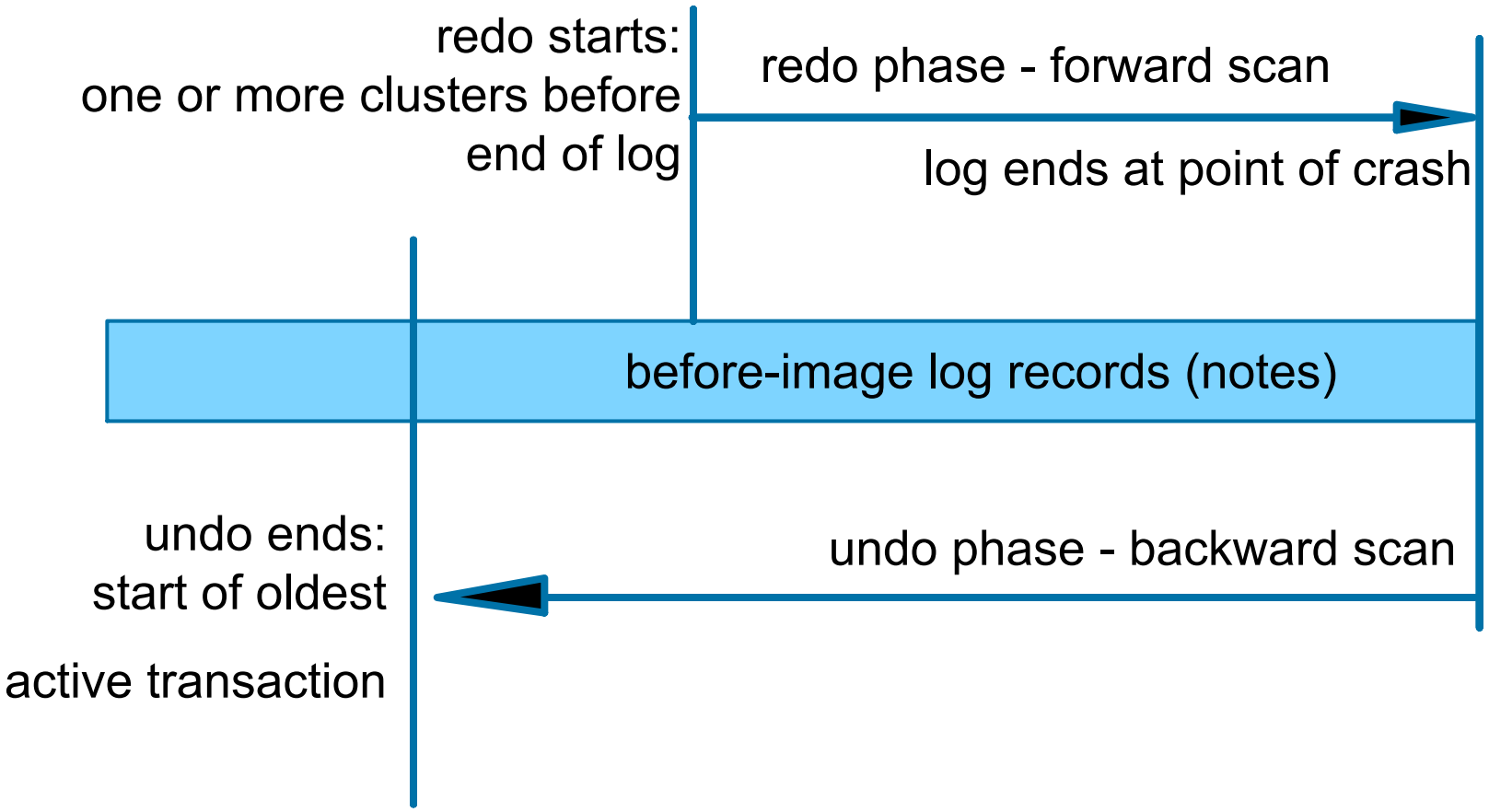
nothing to do
we already have
version 2 of the block

note is skipped

complete database state - 3 parts



crash recovery processing



now we are good.

everything is back the way it was before

you touched it

PROGRESS
software

That's all we have time for today, except

Answers

email:

gus@progress.com