Benchmarking Linux Filesystems for Database Performance Using the 3n+1 Problem

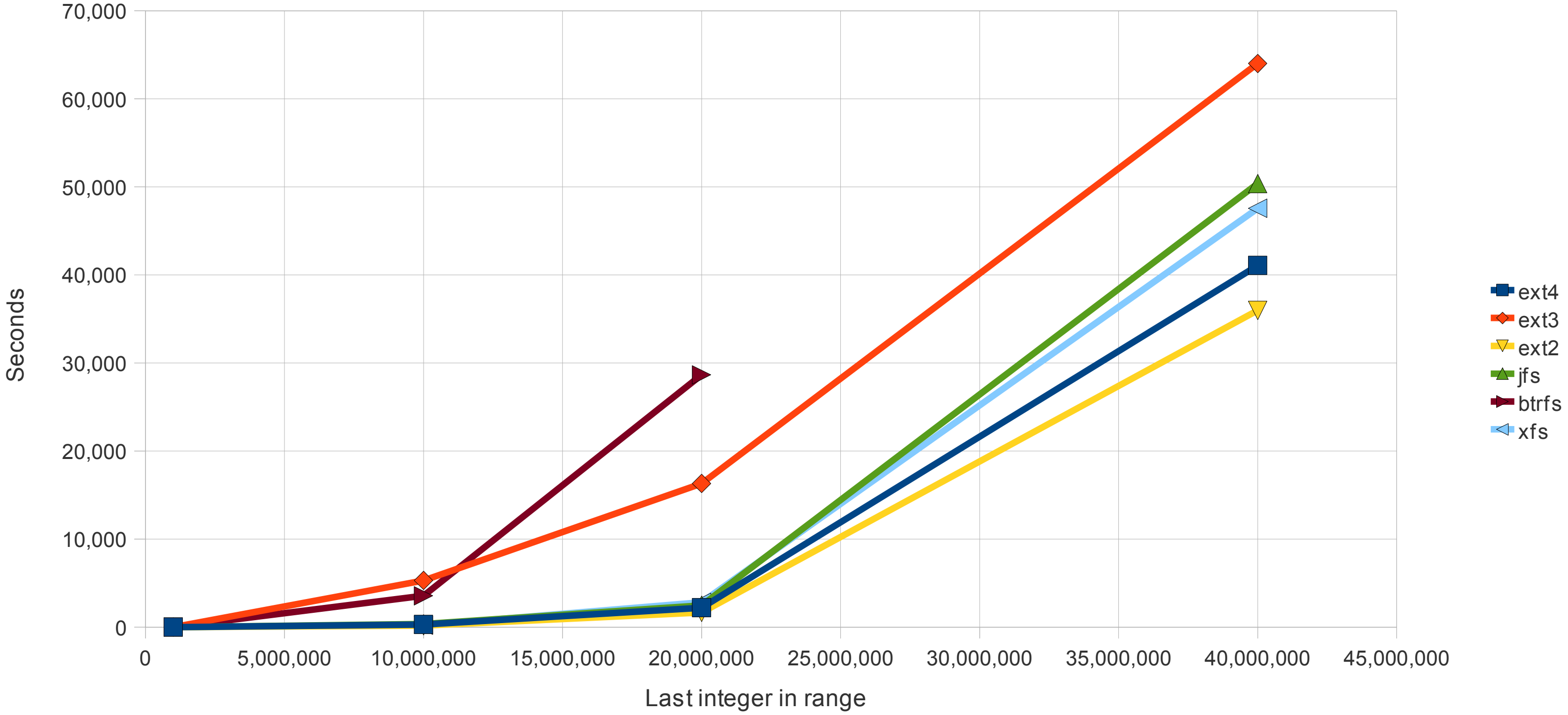
K.S. Bhaskar
SVP, FIS
ks.bhaskar@fisglobal.com
+1 (610) 578-4265

Outline

- Results First!
- Benchmark Workload
- Future Directions

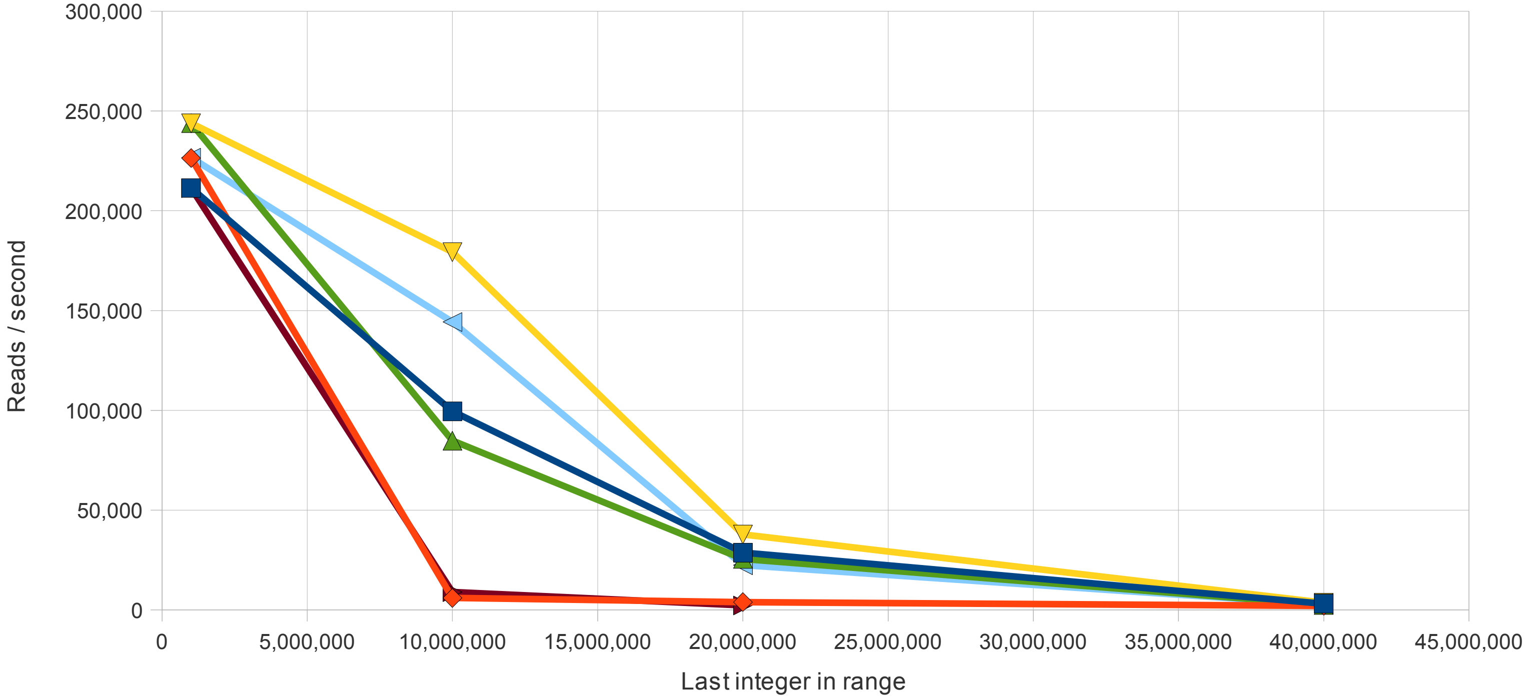
Elapsed Times (seconds)

Finish	ext4	ext3	ext2	jfs	btrfs	xfs
1,000,000	15	14	13	13	15	14
10,000,000	319	5,312	177	374	3,571	220
20,000,000	2,213	16,323	1,680	2,484	28,667	2,834
40,000,000	41,087	64,016	35,989	50,378		47,571



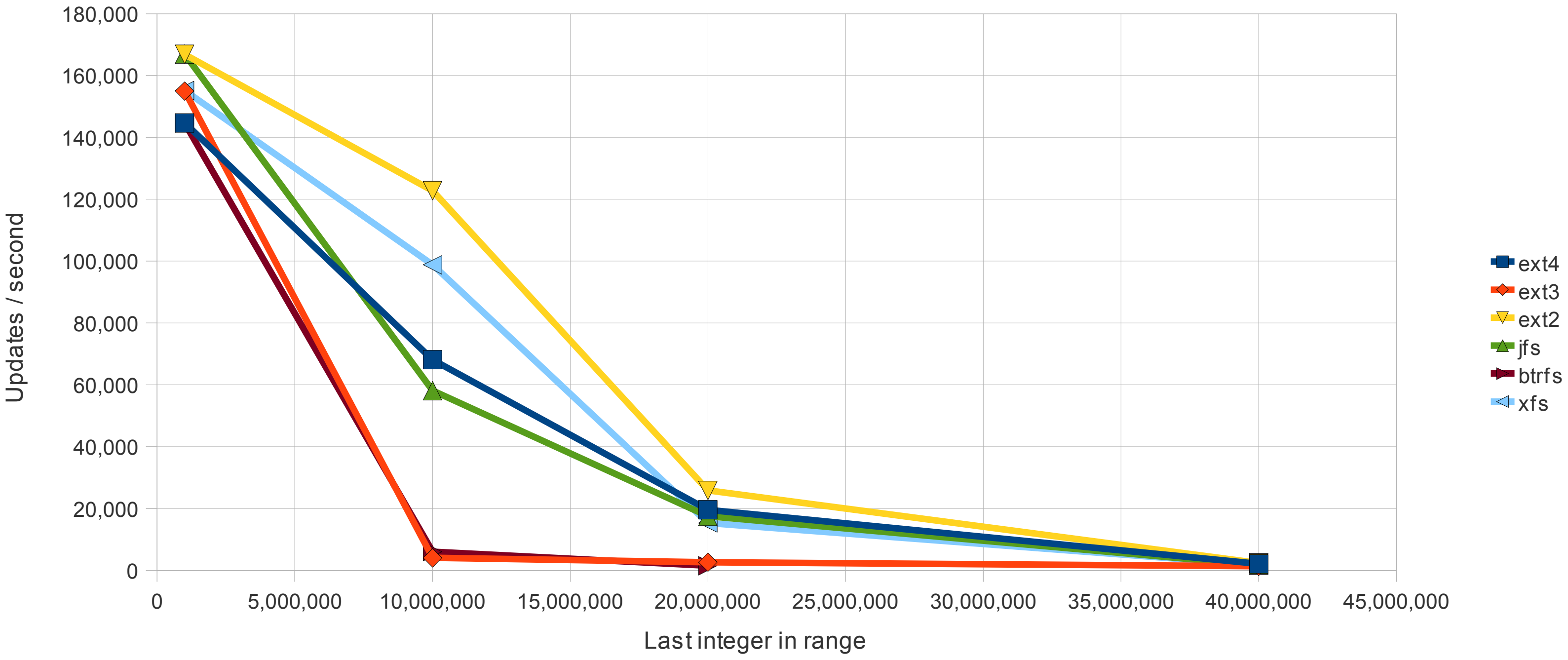
Reads / second

Finish	ext4	ext3	ext2	jfs	btrfs	xfs
1,000,000	211,365	226,421	243,805	243,870	211,341	226,581
10,000,000	99,479	5,974	179,277	84,845	8,886	144,271
20,000,000	28,675	3,888	37,773	25,547	2,214	22,392
40,000,000	3,089	1,983	3,526	2,519		2,668



Updates / second

Finish	ext4	ext3	ext2	jfs	btrfs	xfs
1,000,000	144,698	154,992	166,882	166,947	144,674	155,152
10,000,000	68,131	4,091	122,779	58,107	6,086	98,816
20,000,000	19,638	2,662	25,868	17,495	1,516	15,335
40,000,000	2,115	1,358	2,415	1,725		1,827



Performance Summary

- ext2 is fastest (no surprise – not journaled)
- ext4, jfs and xfs are similar (ext4 has a small edge)
- ext3 is much slower
- btrfs is slowest (no surprise – copy on write has overhead)

Workload Summary

- Database engine specializing in transaction processing applications – FIS GT.M™
  - Database operated with journaling configured for backward recovery as it would be in production
  - Database size grew to 5.3GB
  - Journal size depended on file system, e.g., on jfs, journal files totaled 61.2GB
- Computing problem – length of 3n+1 sequences of integers in a range

3n+1 Problem

- Number of steps that it takes to reach 1:
  - if the number is even, divide it by 2
  - if the number is odd, multiply it by 3 and add 1
- Unproven Collatz conjecture holds number of steps is finite

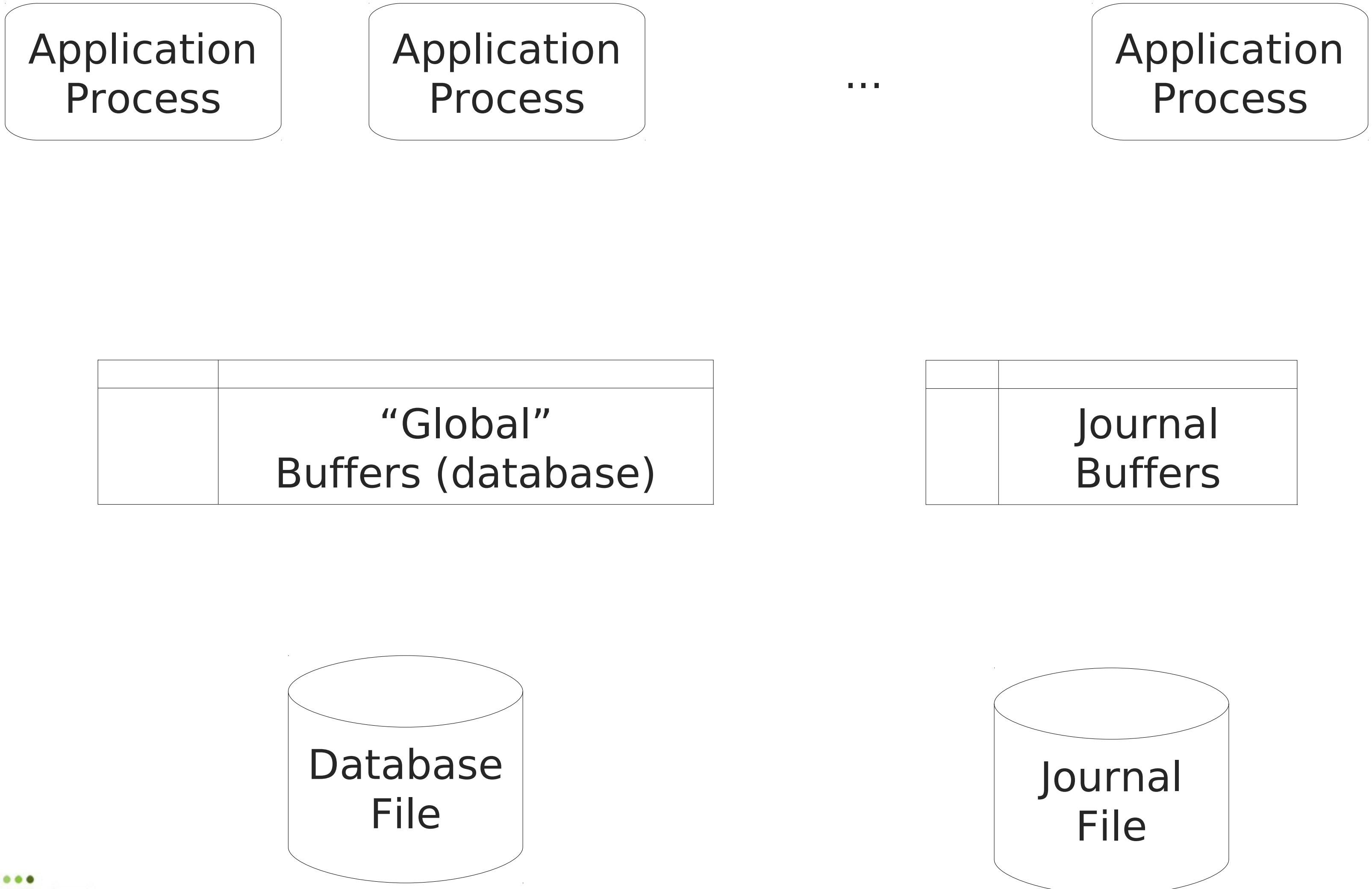
Computer System

- CPU – Quad Core AMD Phenom™ 9850 at 2500MHz
- Cache – L1: 128KBx4; L2: 512KBx4; L3: 2MBx1
- RAM – 4GB DDR2 800
- Disks – Twin Samsung HD103SJ 1TB 3Gbps SATA
  - Four partitions – root occupies one partition on one drive (/dev/sda1)
  - Volume group built from one 705GB partition on each drive (/dev/sda3 and /dev/sdb3)
- OS – 64-bit Ubuntu 10.04.1 (desktop) with kernel 2.6.32-25.
- Benchmark file systems – multiple 100GB file systems, one for each type tested, each striped across the two drives
  - All file systems created with default settings

GT.M Database Engine

- Daemonless architecture
  - Processes cooperate to manage the database
  - Access control via user / group / world permissions
  - Updates go to journal file before they go to database
  - Journal file written to disk before database written
    - Serial access, write only (append) to journal
    - Random access to database
  - Support for ACID transactions
- Largest production databases are a few TB
- Largest production sites serve around 10,000 concurrent users (plus ATM networks, voice response units, etc.)

Database engine



Application Process

Application Process

…

Application Process

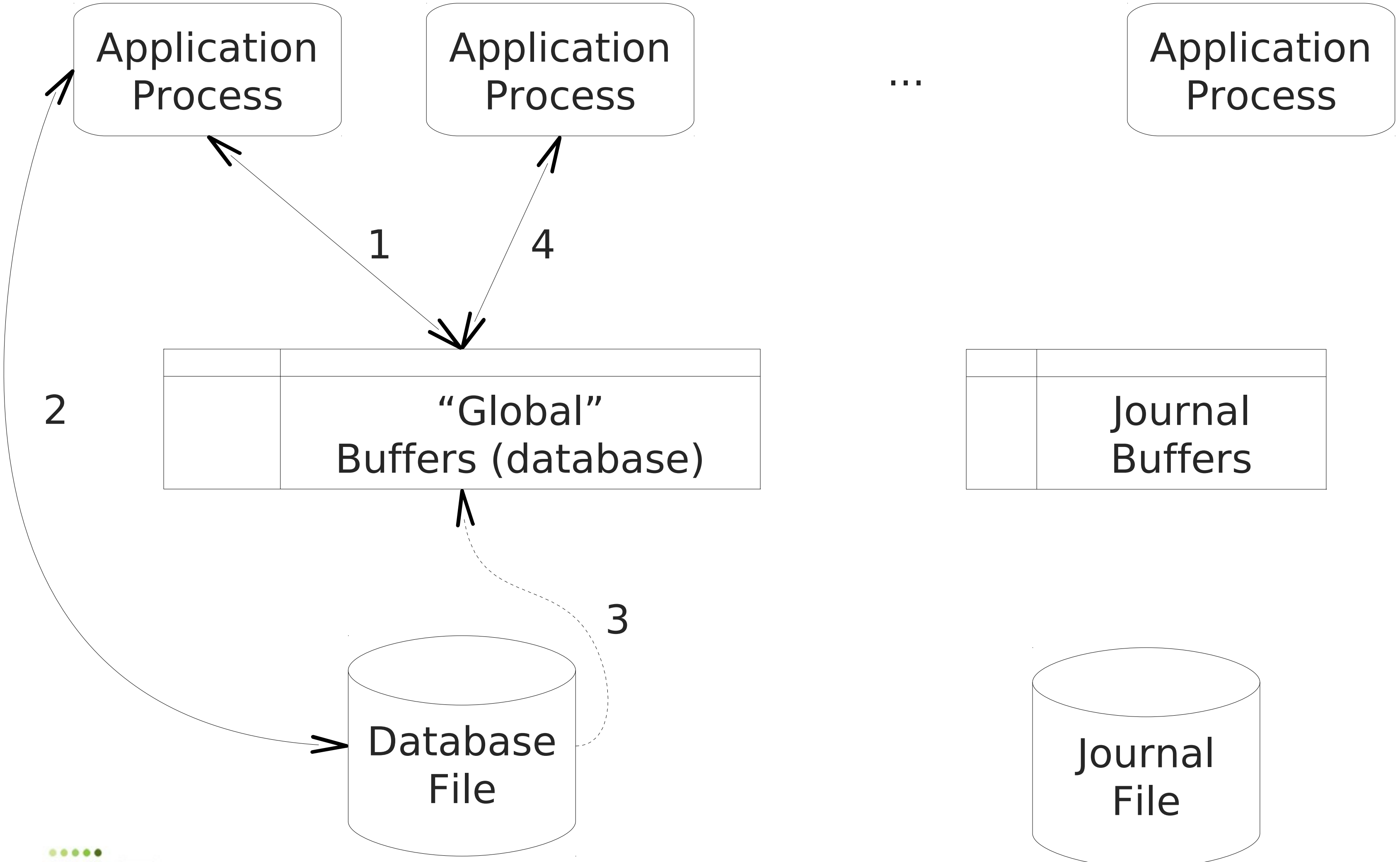
“Global” Buffers (database)

Journal Buffers

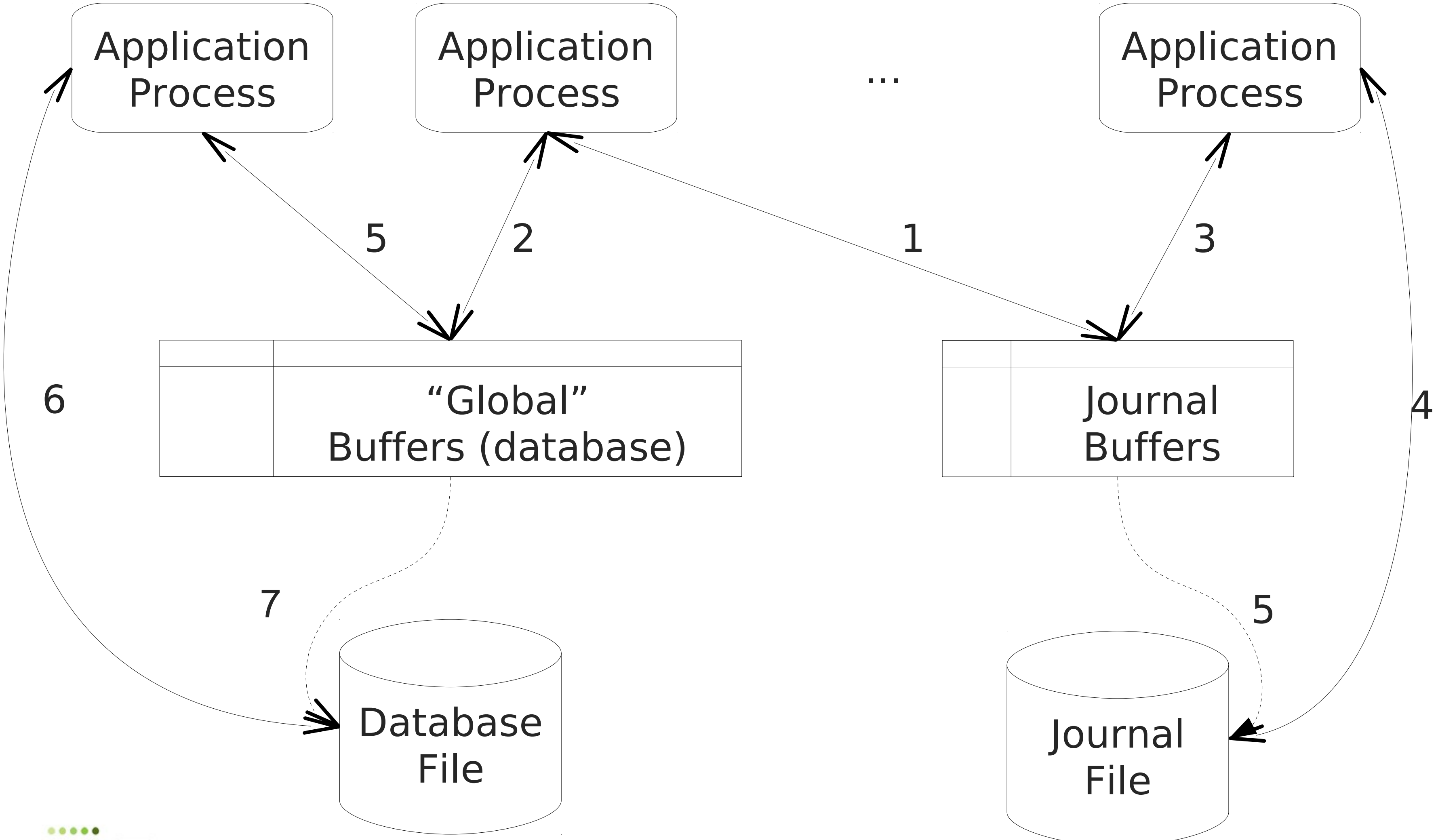
Database File

Journal File

Database engine – reads



Database engine - updates



Access Patterns

- Journal files – sequential write only access
  - Processes use pwrite to write at the end
  - One writer at a time
  - No reads for normal operation (only during recovery)
- Database files – random read-write access
  - Multiple concurrent readers and writers

Performance Revisited

- ext2 is fastest (no surprise – not journaled)
- ext4, jfs and xfs are similar (ext4 has a small edge)
- ext3 is much slower
- btrfs is slowest (no surprise – copy on write has overhead)

- ext4, jfs and xfs seem to be the most interesting

Future Work – IO type

- Other types of IO, e.g., jfs on Magnetic vs. SSD elapsed times
  - (not apples to apples – CPU, RAM and cache differed)

Start	Finish	Magnetic	SSD	Ratio
1	100,000	1	1	1.0
1	1,000,000	13	9	1.4
1	10,000,000	374	193	1.9
1	20,000,000	2,484	631	3.9
1	40,000,000	50,378	2,341	**21.5**

Future Work – DB tuning

- Two tuning parameters – sync_io and $gtm_fullblockwrites (1 to 20,000,000 run)

File system	Sync io	Full block writes	Elapsed times	Relative Speed
jfs	No	No	2,484	100%
jfs	Yes	No	1,733	143%
jfs	No	Yes	2,273	109%
jfs	Yes	Yes	1,743	**143%**
ext4	No	No	2,213	100%
ext4	Yes	No	3,137	71%
ext4	No	Yes	2,122	104%
ext4	Yes	Yes	3,050	73%

Questions / Discussion

K.S. Bhaskar
Senior Vice President, FIS
2 West Liberty Boulevard, Suite 300
Malvern, PA 19355, USA
ks.bhaskar@fisglobal.com
+1 (610) 578-4265

http://fis-gtm.com
http://ksbhaskar.blogspot.com/2010/06/3n1-nosqlkey-valueschema-freesche.html
https://docs.google.com/document/pub?id=1OO2TG4pAuW3MrEPSlzv1zAkIlsADq9PpI46DiIM5lQ8