Repository Structure Considerations for Performance

Perforce Software Performance Lab
Michael Shields, Manager
Tim Brazil, Engineer

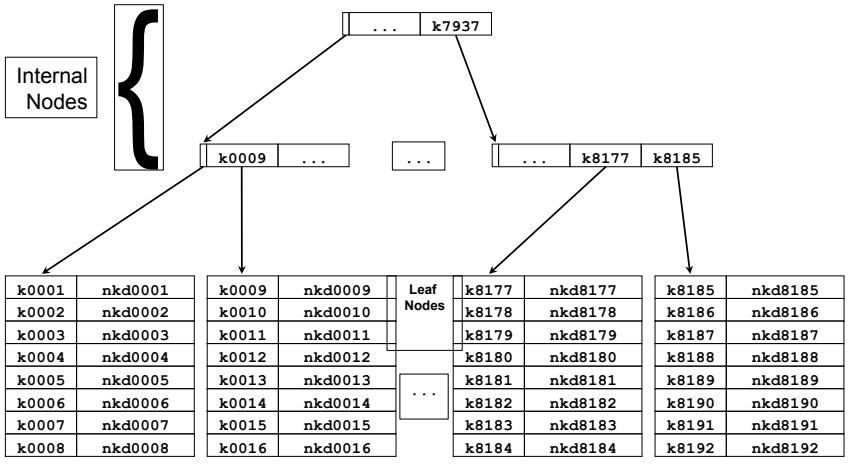
Introduction

- Two of many repository structure decisions
  - path length
  - placement of products and projects
- Perforce benchmarks
- btree internals
- Undocumented and unsupported!
  - subject to change at any time

Perforce db.* files

- Each contains metadata and free page btrees
- Default page size is 8192 bytes
- 16384 bytes for empty (or nearly so) db.* file
  - one empty (or nearly so) leaf node
  - metapage has information about both trees
- Free page btree not created immediately
  - waits until the one leaf node fills

Perforce btree General Structure



Lengthy keys and non-keyed data

- Lengthy keys:
  - decreases number of entries in each node
  - decreases fan-out (referenced by internal nodes)
  - number of internal and leaf nodes increase
  - might cause more levels
- Lengthy non-keyed data
  - decreases number of entries in each leaf node
  - number of internal and leaf nodes increase
  - might cause more levels

Perforce btree Utilities

- p4d -r $P4ROOT -xv -vdb=2

```
...
GetDb db.rev mode 1
Validating db.rev
tree stats:     leafs: 1865619  internal: 23234 free: 0 levels: 4
                items: 72625000 overflow chains: 0      overflow pages: 0
                missing pages: 0        leaf page free space: 2%
                leaf offset sum: 11412075       wrinkle factor: 6.12
                main checksum: 1829071194 alt checksum 0
Unlocking db.rev.
...
```

  - slow: scans internal and leaf nodes

Perforce btree Utilities

- p4 admin dbstat -h

```
internal+leaf 23234+1865619
page size 8k end page 1888854
generation 4 levels 4 fanout 81
ordered leaves: 97%
Checksum 1829071194
 .... : -1000       283
-1000 :  -100         0
 -100 :   -10     22944
  -10 :    -1         0
            1   1819446
    1 :    10         1
   10 :   100     22663
  100 :  1000         0
 1000 :  ....       281
```

  - fast: only scans internal nodes

Keys Containing Paths

db.* file	paths in key
db.archmap	lbrFile, depotFile
db.have	clientFile
db.integed	toFile, fromFile
db.label	depotFile
db.locks	depotFile
db.resolve	toFile, fromFile
db.rev	depotFile
db.revcx	depotFile
db.revdx	depotFile
db.revhx	depotFile
db.revpx	depotFile
db.revsx	depotFile
db.working	clientFile

Non-Key Data Containing Paths

db.* file	paths in non-keyed data
db.change	root
db.protect	depotPath
db.review	depotPath
db.trigger	depotPath
db.view	viewPath, depotPath

Effects of Path Length on Perforce Server

- CPU cycles for string comparisons
  - lengthy leading part of path is bad
    e.g. //TheVerboseProject/MAIN-will-always-build/...
- Additional nodes require disk space and I/O
- Additional memory required for data structures
- Caches are less effective
- Concurrency can be affected

Benchmark Varying Path Length

- Methodology
- Datasets
- Results

branchsubmit Benchmark

- Provides metrics for
  - duration of compute phase for branch
  - duration of commit phase for submit
  - commit rate (files/second)
- Run twice to normalize filesystem cache effects
- Demonstrates effects of path length
- Available for the asking

Test Hardware

p4d server	Make/Model	Dell 2950 PowerEdge III Server
	Processor(s)	(2) Quad Core Xeon(R) CPU X5450 @3.00GHz
	Memory	16 GB
	Local Disk	(4) 146.8 GB 15k SAS
	OS	SUSE Linux Enterprise Server 10 (SP1)
	Kernel	2.6.16.46-0.12-smp
	Release	P4D/LINUX26X86_64/2008.2/190934 (2009/03/05)
db storage	Make/Model	Violin 1010 DRAM Solid State Disk
	Capacity	413GB formatted XFS

Varying Path - Dataset Formula

064

//depot/…/most/sites/have/longer/…

080

//depot/…/most[0…3]/sites[0…3]/have[0…3]/longer[0…3]/…

096

//depot/…/most[0…7]/sites[0…7]/have[0…7]/longer[0…7]/…

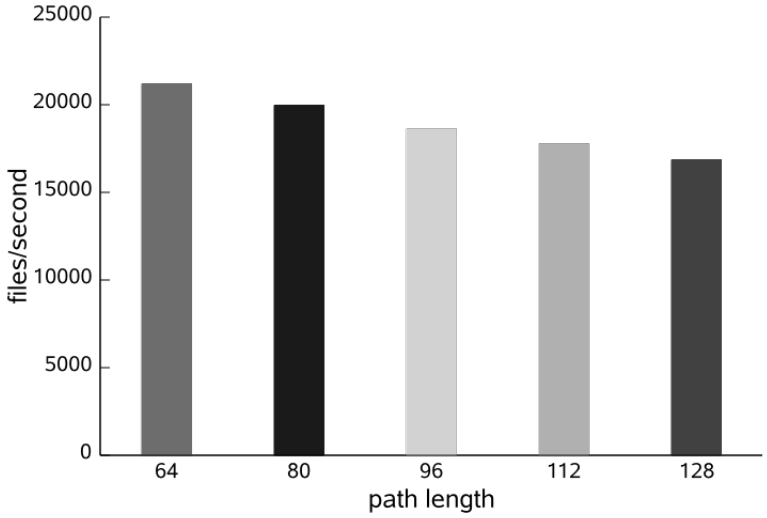
112

//depot/…/most[0…9][A…B]/sites[0…9][A…B]/have[0…9][A…B]/longer[0…9][A…B]/…

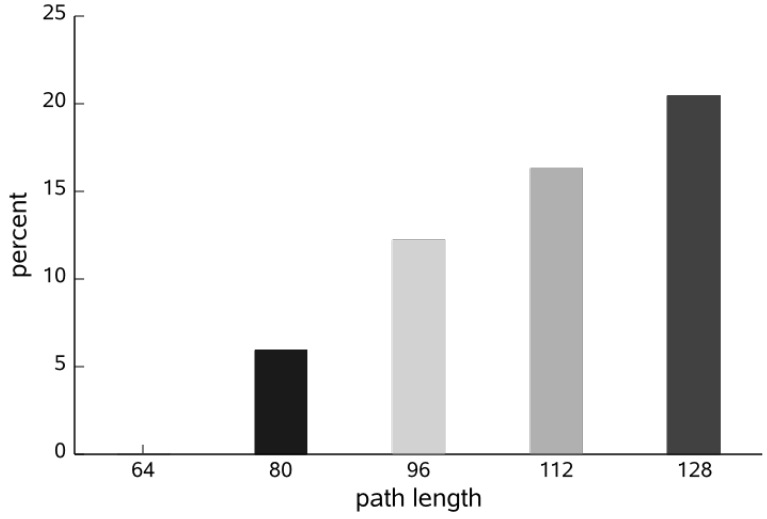
128

//depot/…/most[0…9][A…F]/sites[0…9][A…F]/have[0…9][A…F]/longer[0…9][A…F]/…

Varying Path - Results Chart



Varying Path - Percent Degradation



Varying Path - All Results

Depot	Compute Phase	Commit Duration	Commit Rate
064	4210ms	3305ms	21180 f/s
080	4311ms	3513ms	19925 f/s
096	5117ms	3764ms	18597 f/s
112	5112ms	3947ms	17734 f/s
128	5289ms	4152ms	16859 f/s

Path Length Recommendations

- No need to abandon existing repository
  - just start using shorter paths
  - longer paths only problematic when used
- Shorten lengthy leading part of paths
- Use client mappings for Java and other tools

  e.g. //epiph/MAIN/jsrc/... //my-client/com/our-company/epiphany-project/...
  - Provide defaults and enforce using client triggers

Placement of Products and Projects

- Making each its own depot shortens paths
  - beneficial effect for some btrees
  - db.depot and db.domain increases inconsequential
- Single depot might be problematic
  - if number of first-level directories excessive
- Excessive number of depots might also
  - Server's mapping code working through depot map

Benchmark Varying Depots

- Methodology
- Datasets
- Results

Varying Depots - Advantages?

//productA		//depot/productA
//productB	or	//depot/productB
//productC		//depot/productC

browse Benchmark

- Modeled after typical GUIs (e.g. P4V)
  - repeated dirs, fstat, and filelog commands
    - readonly operations
- Multiple children relentlessly browsing
- Random browsing
  - repeatable using fixed seed
- metadata only
- Available for the asking

Varying Depots - Dataset Formula

//depot[binaryValue]/[binaryValue]top/[main | release]/…

//depot0/1111111111111top/main/most/sites/…/jam # 2 depots

//depot0000000/1111111top/main/most/sites/…/jam # 128 depots

//depot0000000000000/1top/main/most/sites/…/jam # 8192 depots

Varying Depots - Datasets

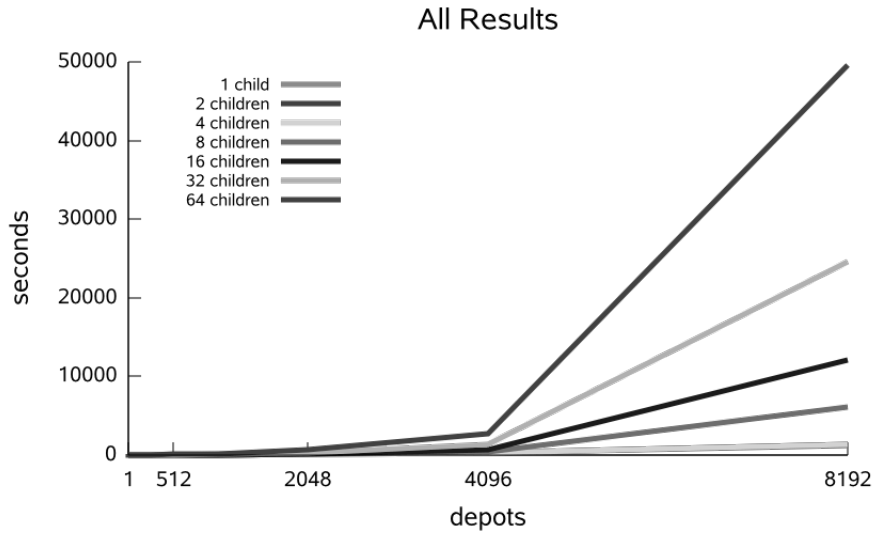
Depots		Directories	
1	depot	00000000000000top…11111111111111top	16384
2	depot0…depot1	0000000000000top…1111111111111top	8192
4	depot00…depot11	000000000000top...111111111111top	4096
8	depot000…depot111	00000000000top...11111111111top	2048
16	depot0000…depot1111	0000000000top...1111111111top	1024
32	depot00000…depot11111	000000000top...111111111top	512
64	depot000000...depot111111	00000000top...11111111top	256
128	depot0000000...depot1111111	0000000top...1111111top	128
256	depot00000000...depot11111111	000000top...111111top	64
512	depot000000000...depot111111111	00000top...11111top	32
1024	depot0000000000...depot1111111111	0000top...1111top	16
2048	depot00000000000...depot11111111111	000top...111top	8
4096	depot000000000000...depot111111111111	00top...11top	4
8192	depot0000000000000...depot1111111111111	0top...1top	2

Varying Depot - Test Scenarios

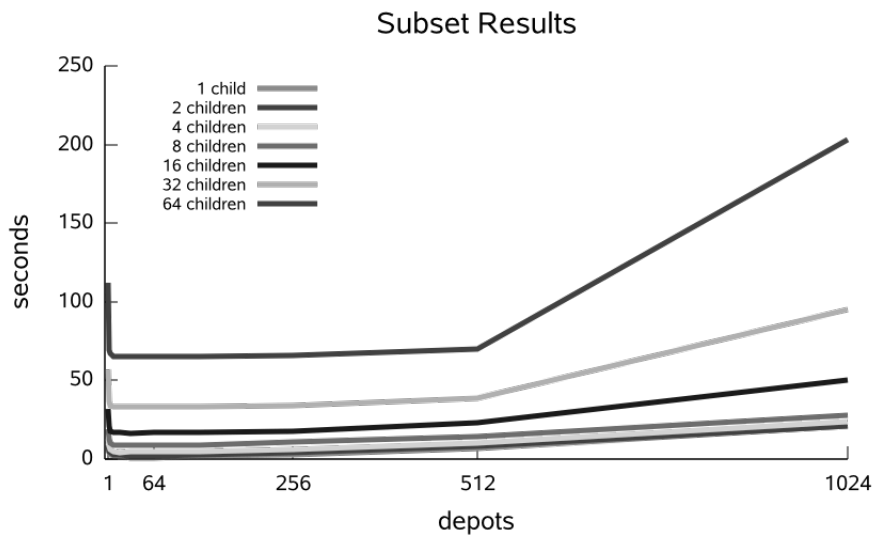
Depots
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192]

Children
[1, 2, 4, 8, 16, 32, 64]

Varying Depots - Results



Varying Depots - Results



Varying Depots - 1 Depot / 1 Child

2009/04/02 14:03:09 pid 2376 … 'user-dirs -C //depot/*'
2009/04/02 14:03:09 pid 2376 **completed .189s** 36+140us 0+0io 0+0net 0k 0pf

2009/04/02 14:03:09 pid 2376 … 'user-fstat -P -C -Olh //depot/*'
2009/04/02 14:03:09 pid 2376 **completed .162s** 20+140us 0+0io 0+0net 0k 1pf

2009/04/02 14:03:09 pid 2376 … 'user-dirs -C //depot/00000001110010top/*'
2009/04/02 14:03:09 pid 2376 **completed .000s** 0+0us 0+0io 0+0net 0k 0pf

2009/04/02 14:03:09 pid 2376 … 'user-fstat -P -C -Olh //depot/00000001110010top/*'
2009/04/02 14:03:09 pid 2376 **completed .000s** 0+0us 0+0io 0+0net 0k 0pf

Varying Depots - 8 Depots / 1 Child

2009/04/02 09:49:13 pid 12752 … 'user-dirs -C //depot101/*'
2009/04/02 09:49:13 pid 12752 **completed .024s** 12+12us 0+0io 0+0net 0k 1pf

2009/04/02 09:49:13 pid 12752 … 'user-fstat -P -C -Olh //depot101/*'
2009/04/02 09:49:13 pid 12752 **completed .021s** 8+12us 0+0io 0+0net 0k 0pf

2009/04/02 09:49:13 pid 12752 … 'user-dirs -C //depot101/00000001110top/*'
2009/04/02 09:49:13 pid 12752 **completed .000s** 4+0us 0+0io 0+0net 0k 0pf

2009/04/02 09:49:13 pid 12752 … 'user-fstat -P -C -Olh //depot101/00000001110top/*'
2009/04/02 09:49:13 pid 12752 **completed .000s** 0+0us 0+0io 0+0net 0k 0pf

Varying Depots - 4096 Depots / 1 Child

2009/04/02 10:06:29 pid 19663 … 'user-dirs -C //depot101100111000/*'
2009/04/02 10:06:31 pid 19663 **completed 1.12s** 1108+12us 0+0io 0+0net 0k 1pf

2009/04/02 10:06:31 pid 19663 …'user-fstat -P -C -Olh //depot101100111000/*'
2009/04/02 10:06:31 pid 19663 **completed .585s** 584+0us 0+0io 0+0net 0k 0pf

2009/04/02 10:06:31 pid 19663 … 'user-dirs -C //depot101100111000/00top/*'
2009/04/02 10:06:32 pid 19663 **completed 1.12s** 1124+0us 0+0io 0+0net 0k 0pf

2009/04/02 10:06:32 pid 19663 … 'user-fstat -P -C -Olh //depot101100111000/00top/*'
2009/04/02 10:06:33 pid 19663 **completed .585s** 581+4us 0+0io 0+0net 0k 0pf

Number of Depots Recommendations

- Active product or project creation
  - make new depots, not first-level directories
  - excessive number of depots might be a problem
    - 500 depots might be excessive

Summary

- Perforce metadata stored in btrees
  - generally very efficient
- Consider shorter paths
  - doubling the path length might result in 20% hit
- Make depots for new products or projects
  - 500 depots might be excessive

Questions?