



# Best Practices with PostgreSQL on Solaris

**Jignesh Shah – Staff Engineer,  
ISV Engineering, Sun Microsystems Inc**

**PostgreSQL East Conference March 2008**



# About Me



- Working with Sun Microsystems for about 7 1/2 years
  - > Primarily responsibility at Sun is to make ISV and Open Source Community Software applications work better on Solaris
- Prior to Sun worked as ERP Consultant
- Worked with various databases (DB2 UDB, PostgreSQL, MySQL, Progress OpenEdge, Oracle)
- Worked with various ERP (QAD, Lawson) and CRM (Dispatch-1), etc
- Previous responsibilities also included : Low Cost BIDW



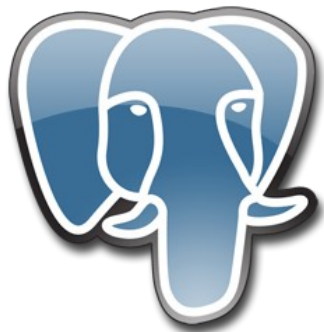
# Agenda

- Deployment tips
  - > Storage
  - > File systems
  - > Solaris tunables
  - > PostgreSQL tunables
- Monitoring
  - > Solaris and other tools
- Summary
- References



# What is not covered

- Solaris Internals
- PostgreSQL Internals
- Replication and Clustering
- High Availability



# Best Practices to deploy PostgreSQL on Solaris



# Solaris 10 or Solaris Express?

- Start with Solaris 10 8/07 or latest Solaris Express
- How to decide whether Solaris 10 8/07 or Solaris Express (or OpenSolaris binary distribution) ?
  - > If long term support from Sun is desired use Solaris 10 8/07
  - > If latest version/feature of Solaris is desired without depending on Support from Sun use latest Solaris Express builds
- Make sure to use at least PostgreSQL 8.2 (and not the default PostgreSQL 8.1 on Solaris 10)
- For 64-bit versions and/or PostgreSQL 8.3 use latest Solaris Express builds



# Understanding PostgreSQL IO

- For simplified view consider three buckets of IO
  - > \$PGDATA – Includes CLOG, control file etc – All 8K,
  - > pg\_xlog, Write Ahead Log: Mostly writes: IO size 8KB-256KB (observed) (Depends on various postgresql.conf tunables)
  - > Relation Files: Data Tables, Index, etc: IO Size 8KB Mix of reads and writes, mix of random,sequential
- Similarly create at least three file systems on Solaris so each file system can be tuned according to the needs of the type of IO
- Relation Files can be easily separated and/or further broken down using TABLESPACES in PostgreSQL



# Setting up IO and UFS Tunable

- By default, UFS aims to cache only small files. Any file bigger than 32KB (like PostgreSQL data files) is generally NOT cached for long.
- By default, on X86 / X64 biggest IO is limited to 56KB by default (compared to 128KB on SPARC)
- By default effective UFS cache is 12% on SPARC (segmap\_percent) and only 64MB on x64 (segmapsize)
- Setting Effective cache higher in PostgreSQL on Solaris **without** tuning UFS caching could hurt performance



# Setting up IO and UFS Tunables

- Increase Maximum IO Size and in effect Readahead for UFS (not in case of directio)
- Set maxphys and klustsize to 1MB
  - > set maxphys=1048576
  - > set klustsize=1048576
- UFS buffer map might need tuning
  - > X86 /X64
    - > set ufs:freebehind=0
    - > set segmapsize=1073741824
  - > SPARC
    - > set freebehind=0
    - > set segmap\_percent=25



# Best Practices for Storage LUNS

- For LUNS exported from external storage arrays, most efficient strip size depends on postgresql.conf parameters. Start with 128K strip size.
  - > If using commit\_delay then higher strip size is preferred (128k-256K) for log filesystem. Recommended to use RAID1 or RAID10. (Avoid RAID-5 for logs)
  - > For other file systems , even though most common IO block size in PostgreSQL is 8K, do not use less than 32KB strip size for layouts. Recommended between 32KB-256KB depending on workload
    - > Sequential scan from database tables and indexes is slow using lower strip size . Bigger strip size impacts response times specially for OLTP Workload.



# Best Practices for Storage/Directory Layout (using UFS)

- Start with three filesystems for PostgreSQL databases
  - > One for \$PGDATA - UFS Buffered (default)
  - > One for \$PGDATA/pg\_xlog – UFS DirectIO (forcedirectio)
  - > One for default tablespace for the database to be created - UFS Directio (forcedirectio,noatime)
    - > (This could require noforcedirectio depending on workload)



# Best Practices for Storage/Directory Layout (using ZFS)

- Separate pool for log filesystem
  - > Mirror device (no RAID-Z) with recordsize 128K
  - > Enable `commit_delay` in `postgresql.conf` to allow logs to write multiple log records in 1 write (bigger than 8K)
- Defaults good for `$PGDATA`
- For default tablespace, defaults probably okay
  - > (but highly workload dependent) Need to figure out the ratio of sequential scans and random I/Os happening first before changing recordsize
- System should not be RAM or CPU bound for ZFS to perform best



# Setting up Resources for User

- Setup resources as follows:

- > `projadd -U pguser user.pguser`
- > `projmod -a -K "project.max-shm-ids=(priv,32768,deny)" user.pguser`
- > `projmod -a -K "project.max-sem-ids=(priv,4096,deny)" user.pguser`
- > `projmod -a -K "project.max-shm-memory=(priv,13589934592,deny)" user.pguser`
- > `projmod -a -K "project.max-msg-ids=(priv,4096,deny)" user.pguser`

- Check `/etc/project`

- > `user.pguser:100::pguser::project.max-msg-ids=(priv,4096,deny);project.max-sem-ids=(priv,4096,deny);project.max-shm-ids=(priv,32768,deny);project.max-shm-memory=(priv,13589934592,deny)`

- Higher limits do not result in wasted memory but are really limited to avoid Denial of Service attacks via user logins



# Best Practices for Solaris

- Use libumem

- > `LD_PRELOAD_32=/usr/lib/libumem.so; export LD_PRELOAD_32`
- > `LD_PRELOAD_64=/usr/lib/64/libumem.so; export LD_PRELOAD_64`
- > `$POSTGRES_HOME/bin/pg_ctl -o -i -D $PGDATA -l $PGDATA/server.log start`

- Use FX Scheduler

- > `/usr/bin/priocntl -s -c FX -i projid 100`



# Best Practices for PostgreSQL.conf

- Use `fdatasync` or `open_datasync` (default)
  - > (Jury divided between the two. I prefer `fdatasync`)
- Enable `commit_delay` for high load environment
  - > (unlike the new `async commit` in 8.3, there is no loss of transactions )
- Increase `checkpoint_segments`
- Increase `wal_buffers`
  - > (when response times are critical with many users)
- Default background writer parameters good in 8.2
  - > Background writer in 8.1 needs tuning



# Best Practices for PostgreSQL.conf

- Shared Bufferpool getting better in 8.2 worth to increase it to 3GB (for 32-bit PostgreSQL) but still not great to increase it more than 10GB (for 64-bit PostgreSQL)
- Temp buffers and work mem are workload dependent
  - > (For high number of connections they have  $x$  max connections impact – sometimes more than connections)
- Maintenance work mem is good between 256MB to 512MB
- (Meant for bigger than 5GB database - Ofcourse)



# Best Practices for Glassfish

- Use Extra JDBC Wrapper driver to cache statements in Glassfish
- Number of steady connections should be increased with caution beyond 100



# Other Tips for Deployment

- If you are using multiple applications or instances use separate userids or projectids to make monitoring easy (Zones/Containers are also another easy options)
- If you plan to use CPU usage “limits” by application (by using Resource Constraints feature of Solaris Containers), allow some time to gather statistical data first on its CPU usage



# Monitoring PostgreSQL On Solaris

Sun Fire X4150 (2 Socket, 8 Cores)

Project user.pguser:

PostgreSQL User

Project user.app

Glassfish App Server

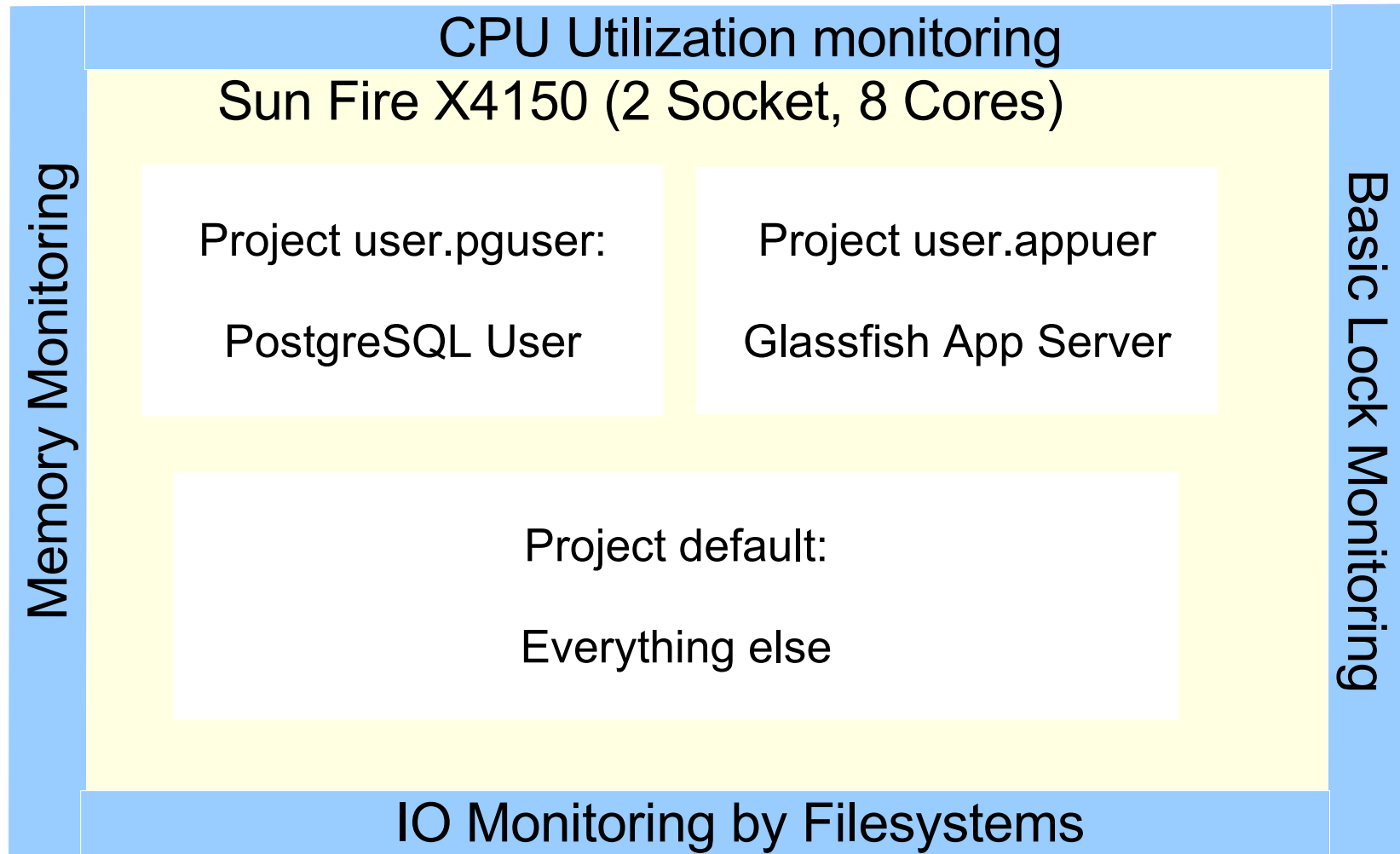
Project default:

Everything else

**Note:** This is just an example and not an actual recommendation for Sun Fire X4150. Actual distribution should consider all applications running on the system.



# Monitoring PostgreSQL On Solaris





# Memory Monitoring

- vmstat 3
  - > Keep an eye on swap/free memory

```
# vmstat 3
kthr      memory          page        disk        faults       cpu
r  b  w    swap  free  re  mf  pi  po  fr  de  sr  s0  s1  s2  s3   in   sy   cs  us  sy  id
0  0  0 1600992 2622832 0   6   0   0   0  0  0  0  0  0  0 1018  287  475  1  1 98
0  0  0 1347744 2400480 31 37  0   0   0  0  0  0  0  0  0 1061 2772  900 23  2 75
0  0  0 1282420 2335524 0   0   0   0   0  0  0  0  0  0  0 1053 2728  899 24  2 74
0  0  0 1217088 2270192 0   0   0   0   0  0  0  0  0  0  0 1057 2653  889 23  2 75
```



# Memory Monitoring

- prstat -a
  - > Memory usage by process (rough aggregation by user)

```
Total: 55 processes, 188 lwps, load averages: 1.02, 0.82, 0.44
  PID USERNAME  SIZE  RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
20787 pguser    3173M 3166M cpu0   0    0   0:08:29  25% postgres/1
20789 root      3332K 2804K cpu3   59    0   0:00:01  0.1% prstat/1
  133 root      4296K 3284K sleep  59    0   0:00:00  0.0% picld/4
  142 root      3536K 2344K sleep  59    0   0:00:00  0.0% devfsadm/6
  290 daemon    2036K 1288K sleep  60   -20   0:00:00  0.0% lockd/2
20772 root      2648K 1716K sleep  59    0   0:00:00  0.0% bash/1
  291 root      2268K 1180K sleep  59    0   0:00:00  0.0% keyserv/3
  138 daemon    3956K 2012K sleep  59    0   0:00:00  0.0% kcfld/3
 1982 root      6888K 3504K sleep  59    0   0:01:53  0.0% nscd/28
  299 root      2132K 1324K sleep  59    0   0:00:00  0.0% ttymon/1
  292 root      1752K  944K sleep  59    0   0:00:01  0.0% sac/1
  348 root      2456K  996K sleep  59    0   0:00:00  0.0% cron/1
  272 daemon    2536K 1348K sleep  59    0   0:01:20  0.0% rpcbind/1
   9 root        10M 9440K sleep  59    0   0:01:22  0.0% svc.configd/17
   7 root        12M  11M sleep  59    0   0:00:41  0.0% svc.startd/13
NPROC USERNAME  SWAP   RSS MEMORY   TIME    CPU
   8 pguser    3175M 3178M   20%   0:28:15  25%
  40 root        68M   76M   0.5%   0:41:52  0.1%
   1 smmsp     1260K 4132K   0.0%   0:00:01  0.0%
   6 daemon     27M   28M   0.2%   0:01:26  0.0%
Total: 55 processes, 188 lwps, load averages: 1.02, 0.83, 0.44
```



# Memory Monitoring

- `pmap -sax $pid`
  - > To understand the details of memory consumption of a process

```
# pmap -sax 20787
20787: /export/home/postgres/pgsql/bin/postgres
      Address      Kbytes      RSS      Anon      Locked  Pgsz  Mode      Mapped File
0000000000400000          4          4          -          -      4K  r-x--  postgres
0000000000401000        340          -          -          -      -  r-x--  postgres
..
0000000000C80000         12         12         12          -      4K  rw---  [ heap ]
0000000000C83000          4          -          -          -      -  rw---  [ heap ]
0000000000C84000          4          4          4          -      4K  rw---  [ heap ]
FFFFFFD7F0000000    120832    120832    120832    120832    2M  rwxsR  [ ism shmId=0x36 ]
FFFFFFD7F0760000    3115440    3115440    3115440    3115440    4K  rwxsR  [ ism shmId=0x36 ]
FFFFFFD7FFF01000         64          8          -          -      -  rwx--  [ anon ]
FFFFFFD7FFF040000         4          4          -          -      -  rwx--  [ anon ]
FFFFFFD7FFF050000        16         16          -          -      4K  r-x--  en_US.ISO8859-1.so.3
...
FFFFFFD7FFDF7000         20         20         20          -      4K  rw---  [ stack ]
FFFFFFD7FFDFC000         12         12          -          -      -  rw---  [ stack ]
FFFFFFD7FFDFF000          4          4          4          -      4K  rw---  [ stack ]
-----
      total Kb      3247600      3245420      3239316      3236272
```



# IO Monitoring

- `iostat -xcznmp 3`
  - > Figure out if IO is distributed well enough on various devices. Keep an eye on %b (busy) and `asvc_t`

```
# iostat -xcznmp 3
```

```
....
```

```
cpu
```

```
us sy wt id  
25 1 0 74
```

```
extended device statistics
```

r/s	w/s	kr/s	kw/s	wait	actv	wsvc_t	asvc_t	%w	%b	device
29.3	0.0	3754.6	0.0	0.7	0.1	23.3	2.2	6	6	c5t5d0
29.3	0.0	3754.6	0.0	0.7	0.1	23.3	2.2	6	6	c5t5d0s0
29.0	0.0	3712.0	0.0	0.7	0.1	23.4	2.2	6	6	c6t5d0s0
28.0	0.0	3584.0	0.0	0.6	0.1	22.4	2.1	6	6	c7t5d0s0
29.0	0.0	3712.0	0.0	0.7	0.1	23.4	2.2	6	6	c4t5d0s0
27.7	0.0	3541.3	0.0	0.6	0.1	22.1	2.1	6	6	c1t5d0s0
27.7	0.0	3541.3	0.0	0.8	0.1	30.5	2.7	7	8	c0t5d0s0
29.0	0.0	3712.0	0.0	0.7	0.1	23.4	2.2	6	6	c6t5d0
28.0	0.0	3584.0	0.0	0.6	0.1	22.4	2.1	6	6	c7t5d0
29.0	0.0	3712.0	0.0	0.7	0.1	23.4	2.2	6	6	c4t5d0
27.7	0.0	3541.3	0.0	0.6	0.1	22.1	2.1	6	6	c1t5d0
27.7	0.0	3541.3	0.0	0.8	0.1	30.5	2.7	7	8	c0t5d0



# IO Monitoring

- fsstat /mountpoints 3
  - > IO by filesystems

```
# fsstat /pgtbs2 3
new  name  name  attr  attr  lookup  rddir  read  read  write  write
file remov chng  get   set   ops    ops   ops  bytes  ops   bytes
68.9K  235    0 28.1M  17   259K   162  362M 2.83T 86.5M 692G /pgtbs2
0      0      0  0      0    0      0  7.72K 61.7M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.65K 61.2M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.55K 60.4M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.72K 61.7M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.64K 61.1M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.38K 59.1M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.58K 60.6M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.59K 60.7M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.61K 60.9M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.65K 61.2M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.68K 61.4M 0      0 /pgtbs2
0      0      0  0      0    0      0  7.59K 60.7M 0      0 /pgtbs2
```



# IO Monitoring

- zpool iostat 3
  - > If ZFS pools are used, then a simplified iostat by pool

```
# zpool iostat 5
```

pool	capacity		operations		bandwidth	
	used	avail	read	write	read	write
pgdata	108G	2.16T	1	4	87.1K	141K
pglog	43.7G	2.22T	0	0	25.4K	52.4K
pgtbs1	24.4G	3.15T	3	0	310K	37.7K
pgtbs2	90.5G	2.63T	9	1	1.13M	95.8K
pgtbs3	93.0G	2.63T	1	0	97.8K	47.2K
pgtbs4	20.5G	2.70T	5	0	620K	45.8K
pgtbs5	13.6G	3.16T	0	0	91.4K	18.6K
pgtbs6	6.12G	1.81T	0	0	66.2K	22.8K
pgdata	108G	2.16T	0	0	0	0
pglog	43.7G	2.22T	0	0	0	0
pgtbs1	24.4G	3.15T	0	0	0	0
pgtbs2	90.5G	2.63T	153	0	19.1M	0
pgtbs3	93.0G	2.63T	0	0	0	0
pgtbs4	20.5G	2.70T	0	0	0	0
pgtbs5	13.6G	3.16T	0	0	0	0
pgtbs6	6.12G	1.81T	0	0	0	0



# CPU Monitoring

- prstat -am / prstat -Jm
  - > CPU utilization by process and aggregation by user/Project id

PID	USERNAME	USR	SYS	TRP	TFL	DFL	LCK	SLP	LAT	VCX	ICX	SCL	SIG	PROCESS/NLWP
20787	pguser	96	2.5	0.0	0.0	0.0	0.0	1.1	0.0	2	62	13K	0	postgres/1
20814	root	0.0	0.7	0.0	0.0	0.0	0.0	99	0.0	21	0	198	0	prstat/1
268	root	0.0	0.1	0.0	0.0	0.0	0.0	100	0.0	3	0	37	0	in.routed/1
16643	pguser	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	68	3	69	0	postgres/1
16644	pguser	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	78	1	94	0	postgres/1
20615	root	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	15	0	30	0	iosum_amd/1
272	daemon	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	1	0	8	0	rpcbind/1
20758	root	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	2	0	16	0	sshd/1
570	root	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	5	0	7	0	snmpd/1
16645	pguser	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	8	0	6	0	postgres/1
9122	root	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	3	0	10	0	sendmail/1
133	root	0.0	0.0	0.0	0.0	0.0	25	75	0.0	0	0	0	0	picld/4
142	root	0.0	0.0	0.0	0.0	0.0	50	50	0.0	0	0	0	0	devfsadm/6
290	daemon	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	0	0	0	0	lockd/2
20772	root	0.0	0.0	0.0	0.0	0.0	0.0	100	0.0	0	0	0	0	bash/1
NPROC	USERNAME	SWAP	RSS	MEMORY	TIME	CPU								
8	pguser	3175M	3178M	20%	0:44:40	25%								
40	root	68M	76M	0.5%	0:41:51	0.1%								
6	daemon	27M	28M	0.2%	0:01:26	0.0%								
1	smmsp	1260K	4132K	0.0%	0:00:01	0.0%								

Total: 55 processes, 188 lwps, load averages: 1.01, 1.00, 0.82



# CPU Monitoring

- mpstat 5
  - > Utilization by CPU useful to see if one of them is more stressed while there are idle CPUs available

```
# mpstat 5
CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
  0     2   0  148   482  263  179   3    20   11   0    94   2   0   0  98
  1     1   0  168   167   3   175   3    19   10   0    77   1   0   0  99
  2     2   0   47   215   62   65   2    13   8    0    63   1   1   0  98
  3     1   0   41   155   6    56   2     7   8    0    54   1   1   0  98
CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
  0     0   0   0    384  248   2    11   1    1   0   2519  94   4   0   2
  1     0   0   13   196   2   331   0    25   9    0    26   0   1   0  99
  2     2   0   93   391  305  266   1    24   8    0    35   0   2   0  98
  3     0   0   0    158   4   235   0    11   5    0    72   2   0   0  98
CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
  0     0   0   0    414  247  184   5    19   6    0   1256  47   2   0  51
  1     0   0   13   93    2   169   6    16   4    0   1148  42   2   0  56
  2     0   0   0    298  230  198   2    23   8    0   113   3   2   0  95
  3     0   0   1    73    4   130   0    11   5    0   129   4   0   0  96
CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
  0     0   0   0  14546  250   82   13   8    5    0   2428  92   6   0   2
  1     0   0  42457  185   2   359   1   32   13   0    14   0   5   0  94
  2     0   0   1  14635  382  326   0   35   16   0    36   0   4   0  96
  3     0   0   17  14332   4   302   1   20   10   0    22   0   2   0  98
```



# Basic Locks Monitoring

- lockstat sleep 5
  - > Kernel level locking profile

```
# lockstat sleep 5
Adaptive mutex spin: 156 events in 5.022 seconds (31 events/sec)
Count indiv cuml rcnt      spin Lock      Caller
-----
  67  43%  43% 0.00          4 0xfffffffffa4af99a8  taskq_thread+0xe3
  22  14%  57% 0.00          4 0xfffffffffa4af99a8  cv_wait+0x70
  ...
-----
Adaptive mutex block: 2 events in 5.022 seconds (0 events/sec)
Count indiv cuml rcnt      nsec Lock      Caller
-----
   1  50%  50% 0.00      8587 0xfffffffffa4af99a8  taskq_dispatch+0x1b8
   1  50% 100% 0.00      6383 0xfffffffffa4af99a8  cv_wait+0x70
-----
Spin lock spin: 21 events in 5.022 seconds (4 events/sec)
Count indiv cuml rcnt      spin Lock      Caller
-----
   7  33%  33% 0.00          6 cpu0_disp      disp_lock_enter+0x1e
   5  24%  57% 0.00          6 cpu0_disp      disp_lock_enter_high+0x9
-----
Thread lock spin: 1 events in 5.022 seconds (0 events/sec)
Count indiv cuml rcnt      spin Lock      Caller
-----
   1 100% 100% 0.00      7123 transition_lock  ts_update_list+0x52
-----
```



# Basic Locks Monitoring

- `plockstat -p $pid`
  - > User Process level locking profile. Useful when `prstat -am` option shows value under LCK

#



# Basic Locks Monitoring

- Dtrace Scripts based on postgresql provider
  - > PostgreSQL level locking profile

```
#!/usr/sbin/dtrace -qs
dtrace:::BEGIN
{
    lckmode[0] = "Exclusive";
    lckmode[1] = "Shared";
}
postgresql*:::lwlock-startwait
{
    self->ts[arg0]=timestamp;
    @count[arg0, lckmode[arg1]] = count();
}
postgresql*:::lwlock-endwait
/self->ts[arg0]/
{
    @time[arg0 ,lckmode[arg1]] = sum (timestamp - self->ts[arg0]);
    self->ts[arg0]=0;
}
dtrace:::END
{
    printf("\n%20s %15s %15s\n", "Lock Id", "Mode", "Count");
    printa("%20d %15s %@15d\n",@count);
    printf("\n%20s %15s %20s\n", "Lock Id","Mode", "Combined Time (ns)");
    printa("%20d %15s %@20d\n",@time);
}
tick-10sec
{
    exit(0);}

# ./lwlock_wait.d
```



# DTrace Monitoring

- To debug specific issues based on outputs of previous tools
- Recommended to use Dtrace Toolkit for easy scripts for most common scripts



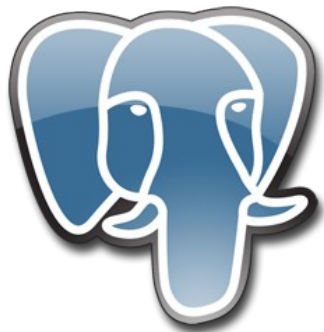
# DTrace Monitoring

- Examples: Scripts from DTrace Toolkit
- <http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/>

```
# ./prustat -p 20787 5
  PID   %CPU   %Mem   %Disk   %Net   COMM
20787  24.52  19.87   0.00   0.00  postgres
  PID   %CPU   %Mem   %Disk   %Net   COMM
20787  24.47  19.87   0.37   0.00  postgres
  PID   %CPU   %Mem   %Disk   %Net   COMM
20787  24.47  19.87   0.34   0.00  postgres
  PID   %CPU   %Mem   %Disk   %Net   COMM
20787  24.50  19.87   0.32   0.00  postgres
```

```
# ./hotuser -p 20787
Sampling... Hit Ctrl-C to end.
^C
```

FUNCTION	COUNT	PCNT
postgres`mdread	1	0.0%
postgres`smgrread	1	0.0%
....		
libc.so.1`memcpy	1789	5.2%
postgres`AllocSetFree	1949	5.7%
postgres`numeric_add	2087	6.1%
postgres`add_abs	2979	8.7%
postgres`AllocSetAlloc	4092	12.0%



# Performance Of PostgreSQL on Solaris



# PostgreSQL on Solaris: Performance

- Published 2 SpecJAppServer2004 result using PostgreSQL 8.2 on Solaris
  - > 778.14 JOPS with Glassfish v1
  - > 813.73 JOPS with Glassfish v2

Mandatory Disclosure:

SPECjAppServer2004 [JOPS@standard](#)

Sun Fire X4200 M2 (4 chips, 8 cores) - 813.73 SPECjAppServer2004 JOPS@Standard

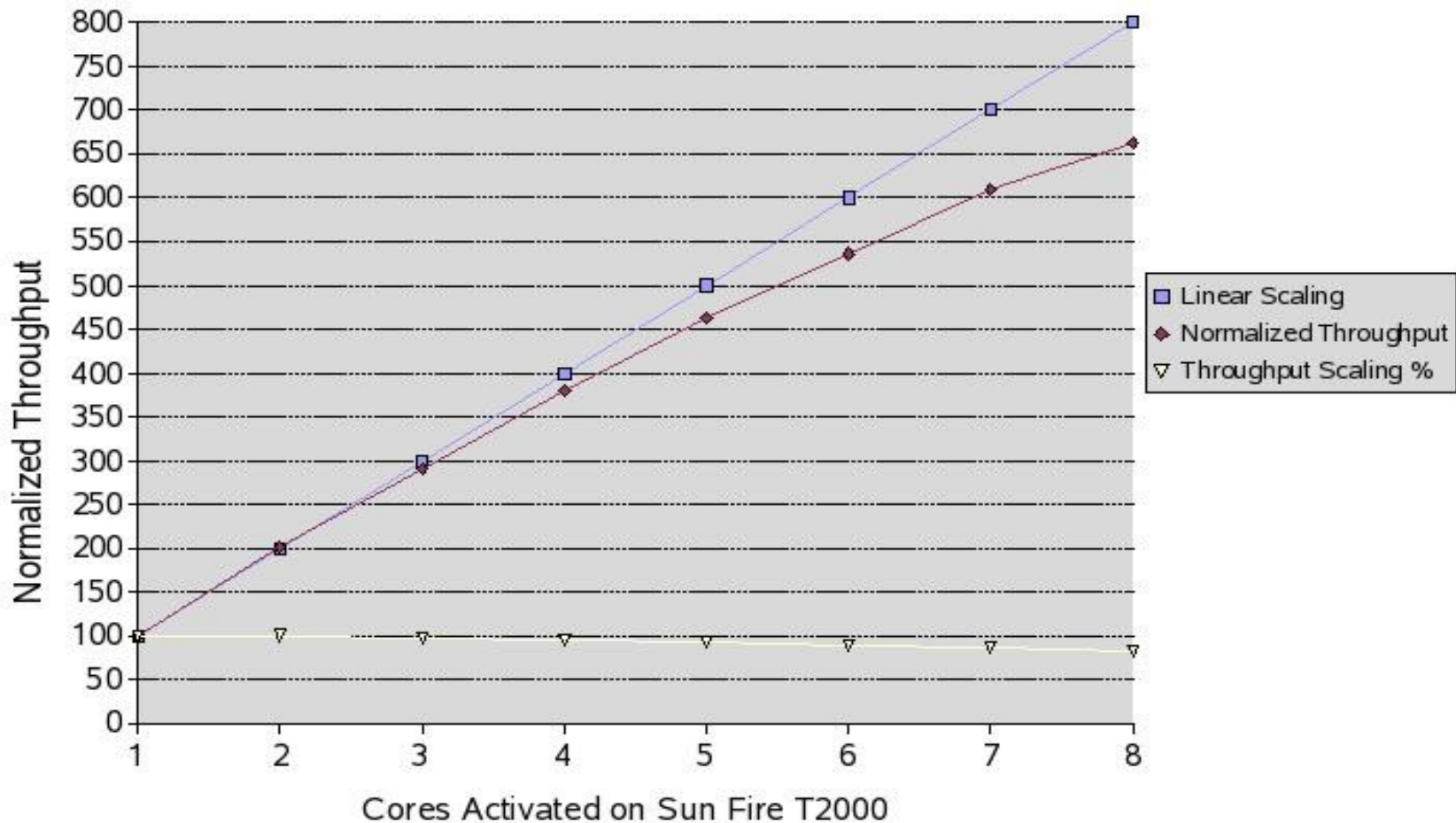
Sun Fire X4200 M2 (6 chips, 12cores) - 778.14 SPECjAppServer2004 JOPS@Standard

SPEC, SPECjAppServer reg tm of Standard Performance Evaluation Corporation. All results from [www.spec.org](http://www.spec.org) as of Jan 8, 2008



# PostgreSQL on Solaris : Scalability

PostgreSQL 8.2.4 Scaling on Sun Fire T2000 with Solaris 10 11/06





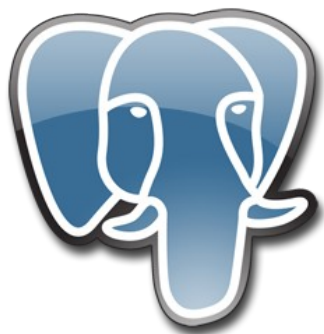
# Summary

- File system layout and tuning key to Performance with PostgreSQL on Solaris
- Make use of default tablespace for database – Easier to Monitor and Tune
- Solaris tools including DTrace are your friends in understanding bottlenecks with PostgreSQL on Solaris

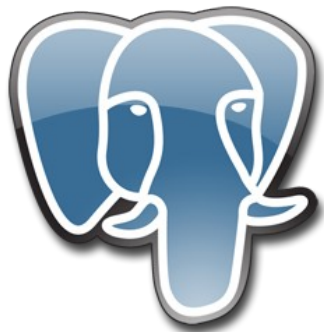


# More Information

- PostgreSQL Question: <postgresql-question@sun.com>
- Blogs on PostgreSQL
  - > Josh Berkus: <http://blogs.ittoolbox.com/database/soup>
  - > Jignesh Shah: <http://blogs.sun.com/jkshah/>
  - > Tom Daly: <http://blogs.sun.com/tomdaly/>
  - > Robert Lor: <http://blogs.sun.com/robertlor/>
- PostgreSQL on Solaris Wiki:  
<http://wikis.sun.com/display/DBonSolaris/PostgreSQL>
- OpenSolaris databases community:  
[databases-discuss@opensolaris.org](mailto:databases-discuss@opensolaris.org)



**Q & A**



# Backup and Additional Information



# PostgreSQL for Solaris

- Integrated into Solaris package
  - > Optimized to perform on Solaris
  - > Incorporated Solaris performance and security features
    - > ZFS
    - > DTrace
    - > Zones
  - > Most reliable and secure DB/OS product
  - > No licensing fees
- Global, 24x7, enterprise-level support from Sun
  - > Service contract available for “PostgreSQL for Solaris”
  - > Community support is also available from PostgreSQL.org



# What's included and what's coming?

- Included
  - > PostgreSQL Core distribution (32-bit)
  - > PostgreSQL JDBC Driver
  - > Most of PostgreSQL Contrib modules
  - > Optional procedural languages PL/pgSQL, PL/perl, PL/Python and PL/Tcl
  - > Supports OpenSSL and Kerberos
- Coming soon:
  - > PgAdmin III GUI for PostgreSQL
  - > PostgreSQL Distribution (64-bit)



# What is in Solaris 10?

- *Solaris 10 6/06 (U2)*
  - > *PostgreSQL 8.1.2 (32-bit)*
- *Solaris 10 11/06 (U3)*
  - > *PostgreSQL 8.1.4 (32-bit)*
- *Solaris 10 8/07 (U4)*
  - > *PostgreSQL 8.1.9 (32-bit) and PostgreSQL 8.2.4 (32-bit)*
  - > *Latest Patch upgrade available to 8.2.6 and 8.1.11 for Solaris 10*
- *Solaris 10 (U5) (Coming Soon)*
  - > *PgAdminIII*



# What is in Solaris Express and OpenSolaris?

- *Solaris Express build 79a (Developer Release)*
  - > PostgreSQL 8.2.5 and 8.1.10
  - > PgAdminIII
- *Solaris Express build 87+*
  - > PostgreSQL 8.3 (32-bit) as well as 8.3 (64-bit)
- *OpenSolaris Developer Preview (Project Indiana )*
  - > Use `pkg(5)` to get PostgreSQL 8.2, 8.1 releases