ORACLE®

Good Practice ?

Dr. Clive King
Solaris Revenue Product Engineering
16th May 2012

clive.king@oracle.com

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Willful blindness ?

- Charles Babbage
  - 1822 Difference Engine
  - 1837 Analytical Engine
- Konrad Zuse
  - 1931 Z1 : 1st electric programmable computer
  - 1942 Z4 : 1st commercial computer
- 1971 Intel introduces 1st microprocessor

- 2012 : Support organisations still see cases on a regular basis where critical data can not be restored from backup

G.P. 1 : don't carry over /etc/system settings

- **Tips to make Solaris slow and unpredictable**
  - set physmem=2097152
  - set kmem_flags=0xf
  - set sq_max_size=20
  - set tune_f_flckrec=640
  - set maxusers=64

- Moral 1 : Don't carry over /etc/system settings from Solaris 2.4, 8 & 9 to Solaris 10 (and beyond)
- Moral 2 : Know what each setting does
- Moral 3 : Application vendor recommended settings are often a historical relic
- *Moral 4: There is no magic*
  - Set performance_problem_fixed=1 /etc/system setting!

G.P. 2 : change control

- Tim Uglow story
- 10's of T2000's ran well for 8 hours then stopped
- Months of admin and support time
- 5 mins to find
  - Heap had been limited to 2GB during testing
  - Carried over to production
- Heap gets to 2GB, process swapped out,
- GC starts – process swapped back in again

G.P. 3 : Be sure what is fact

- Chris Gerhard story
- Long running data corruption issue
- Customer adamant workload drives I/O subsystem flat out
- Failed to reproduce in-house
- Site visit : all columns in iostat are zero
- Disk drives start house keeping when idle for > 90s
  - Disk confused where block to write when I/O re-started
- Fixed in disk firmware
- Spawned Diskomizer

G.P. 4 : Assume the worst

- Rob Hulme story('s)
- Customer A : patched in multi-user -> boot archive corrupt
    - Better alternatives available
- Large US University admissions systems
    - 3rd party driver used unpublished/private kernel interface
    - Interface changed in Kernel Update
    - messy rebuild and restore
    - HP knowledge article noted this [2nd hit on Google]
- Use ZFS snapshot & live upgrade

Patching and upgrade good practice

- Gerry Haskins blogs
- https://blogs.oracle.com/Solaris11Life/ : S11
- https://blogs.oracle.com/patch/ : S10 and below

G.P. 5 : Hassle us to fix sysadmin gotcha's

```
# mdb -kw
  > moddebug/W 1
  > exit
  BOOM !!!
```

```
# echo “moddebug/W 1” | mdb -k
```

Will be fixed in S11u1 and patched in S10

G.P. 6 : Don't forget the firmware

- Memory intensive application very slow on T2000
- Nothing obvious from any *stat tool or compiler tools
- Ran an automated Explorer check
  - Only major item reported was firmware at F.C.S.
- Out of desperation upgraded firmware
- Application ran 4 X faster
- Can tell the same story of T5440
- Firmware release frequency declines over time
- Firmware is **thicker** than it used to be

G.P. 7 : Remember physics

- 3 full config. E25K's destined for a Johanesberg car park on monday morning

- Round trip between 2 zone on same E25K about 50 micro sec.
- Round trip between 2 E25k domains over ethernet about 200 micro-sec.
- Customer architected solution straight into production
- Lots of small packets between COBOL Batch Job and Database
- Batch job 4 times slows on E25K than previous F.J.
  - Customer conclusion : E25K must have slow CPU's

G.P. 8 : Open your mind

- Large UK Bank with SAS risk management application
- Processed many T.B.'s of data in a day
- I/O bound
  - 90% of reads and write to SAS temp files about 500GB
- Which option did they choose?
  - Refresh all of large EMC frame (10 TB) with new : 2 million
  - Put SAS workspace temp directories on fast local storage : 35K
  - Put SAS workspace temp directories in memory : 300k

- Tools did not exist to demonstrate clearly what the working set was

G.P. 8 : G is for governance

- No self respecting techie wants to admit that governance is important
- Framework for decision making and determining who makes it

- sd_max_throttle=20 missing from /etc/system
- Enterprise storage solution disk queues saturated
- Serious performance issues during acceptance testing
  - Roll out delayed

- Technical problem or governance issue?
  - Why was it omitted?

G.P. 9 ZFS

- set zfs_arc_max=0x??????? (in some cases, generally not)
  - echo “::memstat” | mdb -k or echo “::arc” | mdb -k
- Recordsize property – match blocksize

```
dtrace -n 'syscall::*read:entry,syscall::*write:entry
  /fds[arg0].fi_fs == "zfs"/ { @[probefunc] = quantize(arg2); }'
  -n tick-60s'{exit(0)}'
```

- RaidZ[1,2,3] : large sequential I/O only
- Don't run perf. critical filesystems > ~80% full
- Understand what the ZFS intent log really is
- Understand workload profile
  - Sync vs async
  - Io size
  - Random vs sequential

Observing - iostat for the informed !

Ignore!

```
v4v-t5440b-gmp03(5.10)$ iostat -xnzM 1
                extended device statistics
r/s    w/s   Mr/s   Mw/s wait actv wsvc_t asvc_t  %w  %b device
5.0  102.0    0.0   64.0  0.0 12.8    0.0  119.7   0  100 c0t1d0
                extended device statistics
r/s    w/s   Mr/s   Mw/s wait actv wsvc_t asvc_t  %w  %b device
3.0  108.0    0.0   62.3  0.0 15.5    0.0  139.8   0 100 c0t1d0
                extended device statistics
r/s    w/s   Mr/s   Mw/s wait actv wsvc_t asvc_t  %w  %b device
3.0  109.0    0.0   17.1  0.0  6.3    0.0   30.1   0  35 c0t1d0
```

Observing – What's my I/O doing?

- dtrace -n syscall:::entry'{@[probefunc] = count()}'
- dtrace -n syscall::*mmap*:entry'{@[execname] = count()}'
- dtrace -n syscall::read:entry,syscall::*write:entry' {@[probefunc, execname] = quantize(arg2)}'
- dtrace -n io:::start'{@[args[0]->b_flags & O_READ ? “R” : “W”] = quantize(args[0]->b_bcount)}'
- dtrace -n io:::start'{@[fds[arg2]->fi_pathname] = sum(args[0]->b_bcount)}'

Perfect user space memory allocator

- Space efficient for all sizes of allocations
- Fast constant time allocation
- Per cpu caches for all allocation sizes
- Does no fragment under any workload
- Returns unused memory to the OS
- No thread lock contention
- Deals with legacy coding issues such as double free

The perfect allocator can not exist

Choice of User Land Memory Allocators

- Use the right allocator for the job:
  - libc – compromise of performance / space utlisation
  - libmalloc(3LIB) – space efficient /OK performance
  - libbsdmalloc(3LIB) – good perfs / space-inefficient
  - libmapmalloc(3LIB) – returns memory to OS
  - libmtmalloc(3LIB) – MT warm (recent improvements)
  - **libumem**(3LIB) – Fast. MT, can be space efficient

- libc-malloc, bsdmalloc, libmalloc → *single threaded*
- libumem or mtmalloc → *multi-threaded hot*

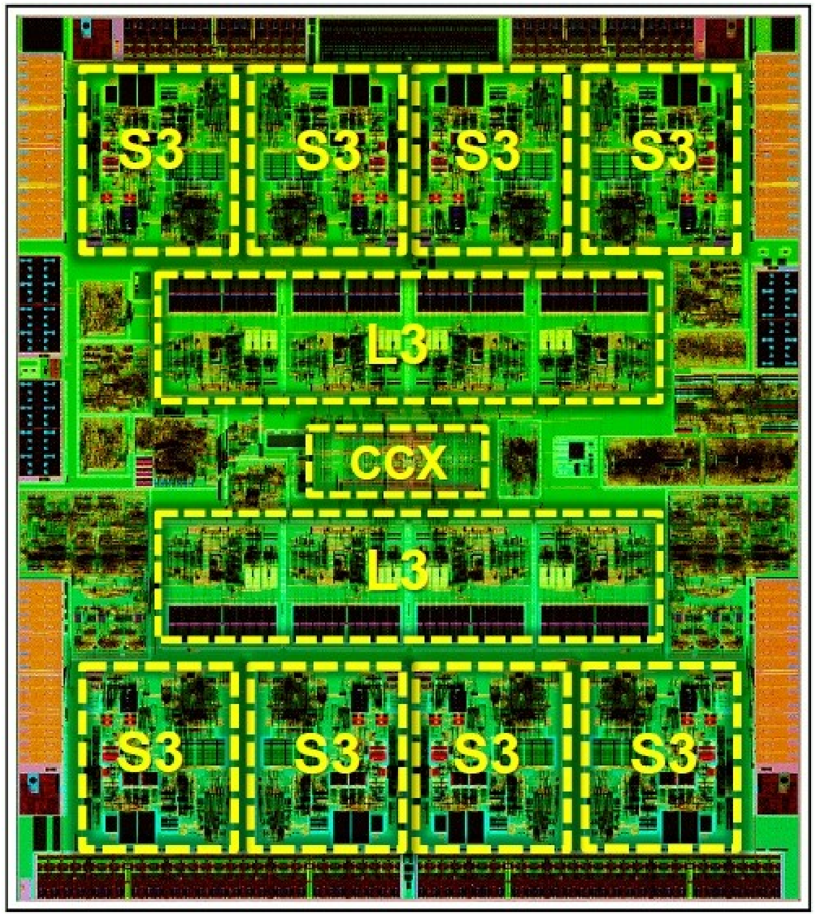
Wrong allocator ! How would I know?

- Go look ……
- prstat -m
  - LCK column is significant
- plockstat -e 30 -s 50 -p >pid<
  - Look for malloc as primary element in stack
- `dtrace -n pid???::malloc:entry'{ @[execname] = quantize(arg0)}'`
  - gives distribution of allocation sizes
  - Add calloc, realloc, valloc

Memory Fragmentation

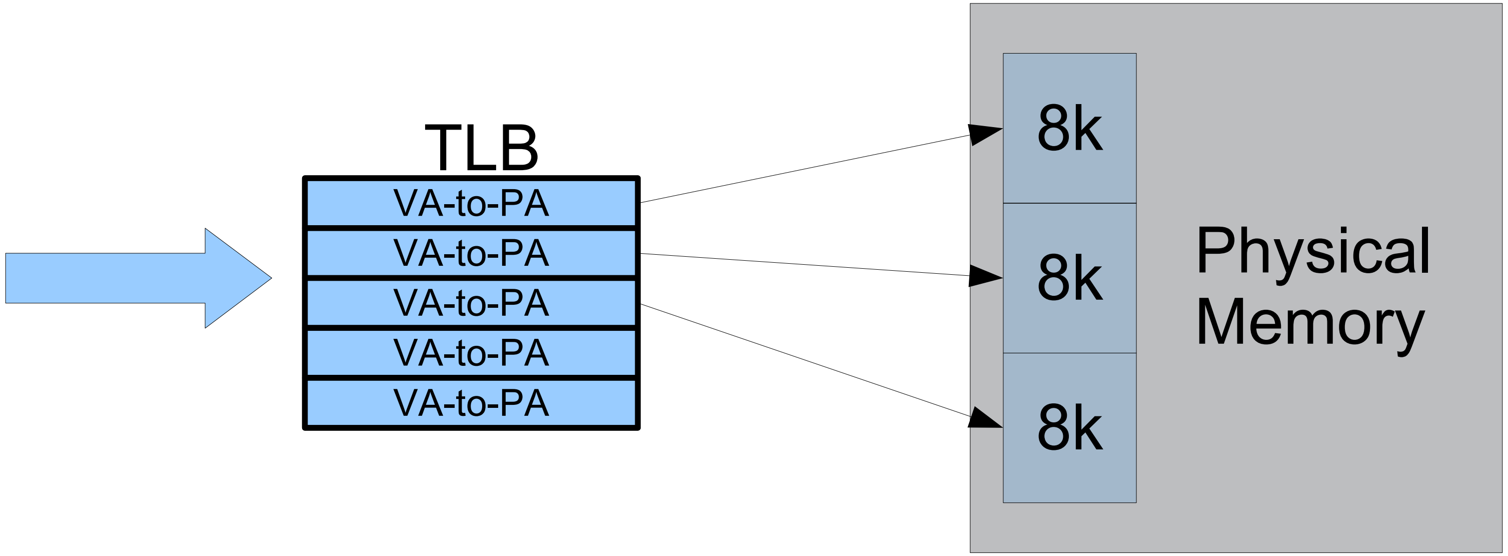
- No allocator can avoid worst case memory fragmentation
  - Worst case is different for each allocator
- For frequent large allocations (16k+)
    - libumem oversize becomes single threaded
    - mmap and munmap
    - umem_cache_alloc
  - malloctl for libmtmalloc
    - MTCHUNKSIZE : fewer calls to brk
  - Avoid realloc
    - different behaviors for libc & libumem
- Avoiding fragmentation is the applications responsibility

What is going on inside ?



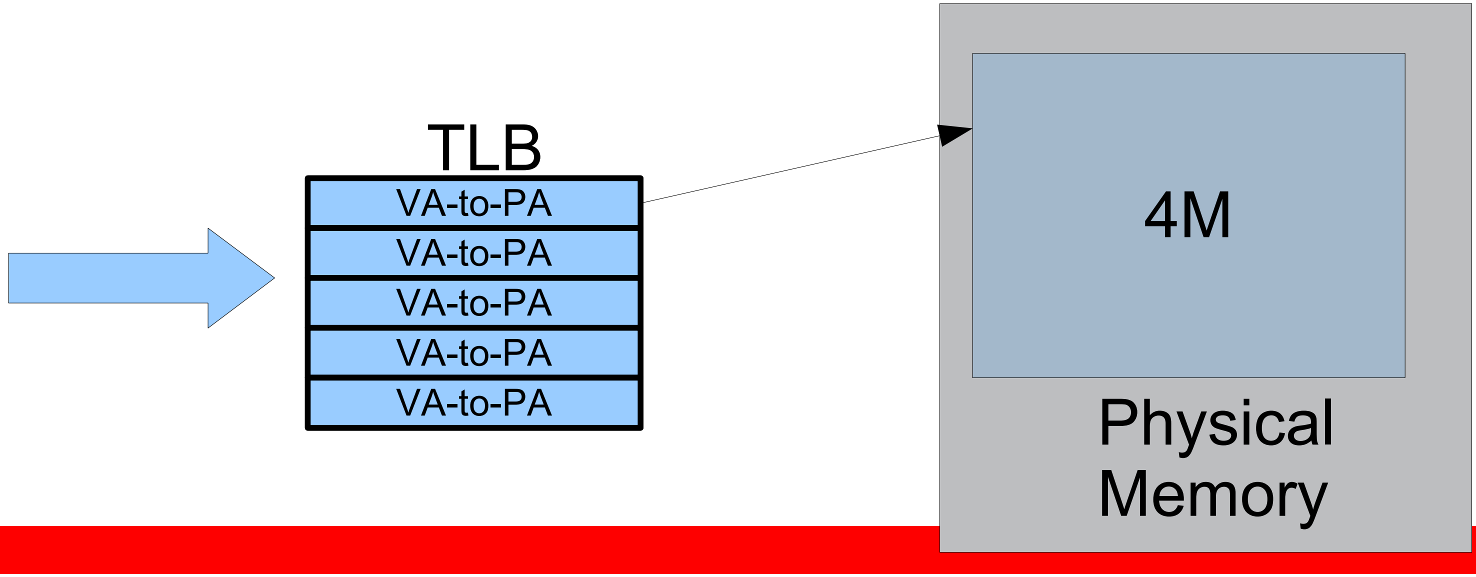
Why Large Pages?

address references
from running threads



512 8k pages for a 4MB segment, versus one 4MB page

address references
from running threads



Observing MMU traps

```
# trapstat -T 1 111
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|        30  0.0         0  0.0 |   2170236 46.1         0  0.0 |46.1
  0 u  64k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 u 512k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 u   4m|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
- - - - - + - - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - - - + - -
  0 k   8k|         1  0.0         0  0.0 |      4174  0.1        10  0.0 | 0.1
  0 k  64k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 k 512k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 k   4m|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      Ttl |        31  0.0         0  0.0 |   2174410 46.2        10  0.0 |46.2
```

This application **could** potentially run 2x faster using large pages!

Performance Instrumentation Counters

- Enable us to track low level events happening on the CPU
  - 80/90% performance issues resolved by standard tools
  - 10% of performance cases require further drill down

- Cpustat(1M) – entire system or cputrack(1M) – per process
    - Generic Events [generic_events(3CPC)]
    - Platform Specific Events, use cpustat -h to find out what's available

- DTrace CPC Provider
  - Generic L2 data cache miss for http executable
    - L2 cache miss → going to RAM which is expensive!

```
dtrace -n 'cpc:::PAPI_l2_dcm-all-10000 /execname == "httpd"/
{@[ufunc(arg1)] = count();}'
```

CPC Counters meet DTrace

Solaris 8   : CPC [Cpu Performance Counter]

Solaris 11 : DTrace CPC provider

PAPI (Performance Application Prog. Interface)

- Cpustat -h
  - Generic Events [generic_events(3CPC)]
  - Platform Specific Events
- PAPI_l2_dcr : Level 2 data cache read
  - AMD 0xF & 0x10 processor : DC+refill_from_L2
  - Intel Pentium Pro : l2_ld
  - US III/IIIi/IV, USIV+ : No

Calculating CPI

- PAPI_tot_cyc
  - AMD Opteron : BU_cpc_clk_unhalted
  - Intel Pentium IV : global_power_events
  - Intel Pentum Pro : cpu_clk_unhalted
  - US I/II/III/IIIi/IV/IV+ : Cycle_cnt

- PAPI_tot_ins
  - Opteron : FR_retired_x86_instr_w_excp_intr
  - Pentium : instr_retired
  - US III/III+ : instr_cnt
  - US IV/IV+ : instruction_counts
  - US T2 : Instr_cnt

- Example

```
root@x4640-tvp540-b:~# cpustat -nc pic0=BU_cpu_clk_unhalted,pic1=FR_retired_x86_instr_w_excp_intr 10 1 | awk '{ printf "%s %.2f cpi\n",$0,$4/$5; }'
10.001   0  tick  18895367  35135759  0.54 cpi
10.002   1  tick 243526553 235456833  1.03 cpi
10.002   2  tick    211986     65399  3.24 cpi
10.003   3  tick   8560908   2016222  4.25 cpi
<snip>
```

What does CPI tells us?

- If CPI is low
  - Examine the application for unnecessary CPU work
  - Get faster CPUs
  - Get more CPUs
- If CPI is high
  - Examine application for unnecessary memory work
  - Recompile with optimization with Oracle C compiler
  - Processor sets to improve memory locality (maybe?)
  - Get CPUs with larger caches
  - Test different CPU architectures (multi-core/multi-threaded)

Questions