Trace File Event Timeline

Presentation to
OAUG Database SIG

Andy Rivenes
Oracle Architect

October 11, 2009

LLNL-PRES-417271

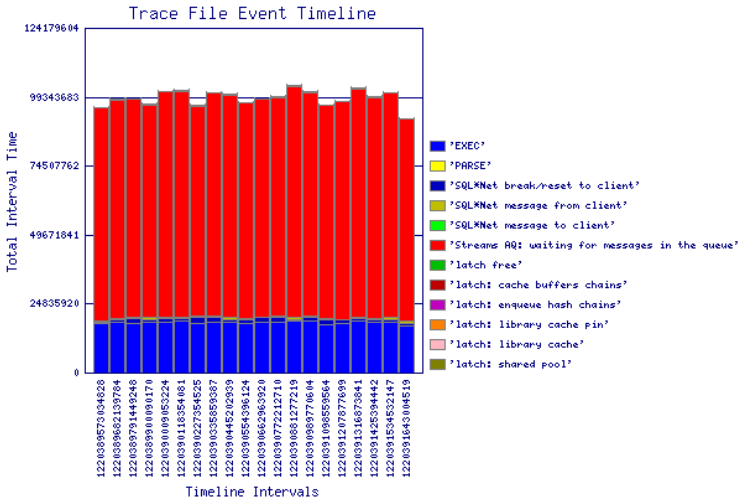
This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

Trace File Event Timeline

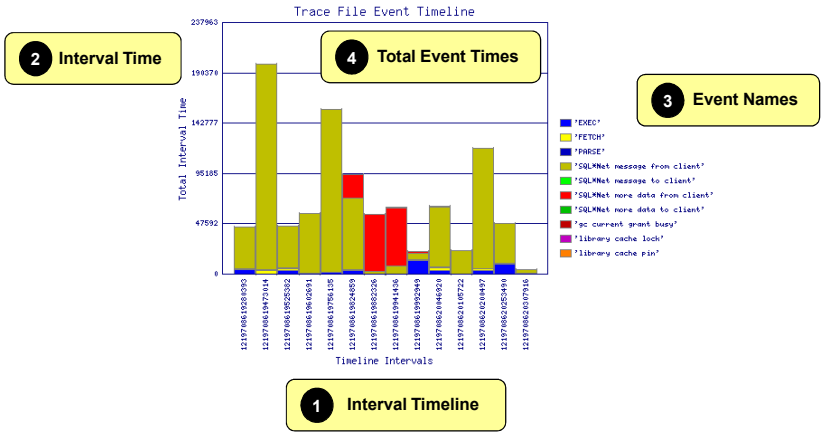
The National Ignition Facility

- Displays event totals in a timeline as they occurred
- Makes it easier to determine roughly when actions took place
- Divided into “approximately” equal interval times
- As a visualization tool it can be displayed as a stacked bar chart

Trace File Event Timeline Graph



How to Read The Graph



Creating A Trace File Event Timeline

The National Ignition Facility

- Slicing the trace file into intervals
- Uniform or non-uniform intervals?
- Reconciling recursive SQL
- Initial attempt:
  - Used non-uniform intervals
  - Divided the trace file time by 20 and used that number as a target interval time
  - Looked at boundaries between WAIT events and recursive depth 0 PARSE, EXEC and FETCH statements

Trace File Review

The National Ignition Facility

```
EXEC #6:c=0,e=83,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1219708745416872
WAIT #6: nam='SQL*Net message to client' ela= 1 driver #bytes=1 p3=0 obj#=-1 tim=1219708745416904
FETCH #6:c=0,e=84,p=0,cr=8,cu=0,mis=0,r=1,dep=0,og=1,tim=1219708745417024
STAT #6 cnt=1 pid=0 pos=1 obj=0 op='NESTED LOOPS  (cr=8 pr=0 pw=0 time=0 us cost=5 size=89
  card=1)'
STAT #6 cnt=1 pid=1 pos=1 obj=71003 op='TABLE ACCESS BY INDEX ROWID ODM_PUBLICOBJECT (cr=4 pr=0
  pw=0 time=0 us cost=3 size=73 card=1)'
STAT #6 cnt=1 pid=2 pos=1 obj=71009 op='INDEX UNIQUE SCAN SYS_C009848 (cr=3 pr=0 pw=0 time=0 us
  cost=2 size=0 card=1)'
STAT #6 cnt=1 pid=1 pos=2 obj=70836 op='TABLE ACCESS BY INDEX ROWID ODM_DOCUMENT (cr=4 pr=0 pw=0
  time=0 us cost=2 size=16 card=1)'
STAT #6 cnt=1 pid=4 pos=1 obj=70839 op='INDEX UNIQUE SCAN SYS_C009781 (cr=3 pr=0 pw=0 time=0 us
  cost=1 size=0 card=1)'
WAIT #6: nam='SQL*Net message from client' ela= 1202 driver #bytes=1 p3=0 obj#=-1
  tim=1219708745418336
FETCH #6:c=0,e=2,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=1219708745418358
WAIT #6: nam='SQL*Net message to client' ela= 1 driver #bytes=1 p3=0 obj#=-1 tim=1219708745418375
WAIT #6: nam='SQL*Net message from client' ela= 87423 driver #bytes=1 p3=0 obj#=-1
  tim=1219708745505813
EXEC #3:c=0,e=84,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1219708745505944
WAIT #3: nam='SQL*Net message to client' ela= 1 driver #bytes=1 p3=0 obj#=-1 tim=1219708745506104
FETCH #3:c=0,e=174,p=0,cr=11,cu=0,mis=0,r=3,dep=0,og=1,tim=1219708745506134
WAIT #3: nam='SQL*Net message from client' ela= 2607 driver #bytes=1 p3=0 obj#=-1
  tim=1219708745508784
```

Creating A Trace File Event Timeline

The National Ignition Facility

- Second attempt will:
  - Allow boundaries to fall between recursive SQL (i.e. recursive depth greater than 0)
  - This will require tracking all “child” time in order to attribute “interval” PARSE, EXEC and FETCH time for each interval
  - Should provide a much better picture for highly recursive workloads and more accurate breakdown of CPU usage

Recursive SQL Relationships

The National Ignition Facility

```
  Line 39: EXEC #4:c=0,e=626,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394068956393,hv=1020616855
  Line 40: FETCH #4:c=69989,e=68532,p=0,cr=630,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069024985,hv=1020616855
  Line 42: EXEC #4:c=0,e=317,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069173174,hv=1020616855
  Line 43: FETCH #4:c=71990,e=70466,p=0,cr=630,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069243668,hv=1020616855
PARSE: cpu=          0 ela=          0
EXEC:  cpu=          0 ela=        943
FETCH: cpu=     141979 ela=     138998

Recursive: cpu=     141979 ela=     139941; Exclusive: cpu=    -141979 ela=    -139903
Line 53: PARSE #6:c=0,e=38,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1220394069245479,hv=562127231

  Line 54: EXEC #4:c=0,e=284,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069247543,hv=1020616855
  Line 55: FETCH #4:c=73988,e=72207,p=0,cr=630,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069319777,hv=1020616855
  Line 56: EXEC #4:c=0,e=291,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069320320,hv=1020616855
  Line 58: FETCH #4:c=71989,e=70324,p=0,cr=630,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394069390667,hv=1020616855
  Line 60: EXEC #4:c=0,e=322,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394070369487,hv=1020616855
  Line 61: FETCH #4:c=71990,e=70318,p=0,cr=630,cu=0,mis=0,r=0,dep=1,og=1,tim=1220394070439833,hv=1020616855
PARSE: cpu=          0 ela=          0
EXEC:  cpu=          0 ela=        897
FETCH: cpu=     217967 ela=     212849

Recursive: cpu=     217967 ela=     213746; Exclusive: cpu=       2999 ela=     980777
Line 62: EXEC #6:c=220966,e=1194523,p=0,cr=1896,cu=0,mis=0,r=0,dep=0,og=1,tim=1220394070440145,hv=562127231

PARSE: cpu=          0 ela=         38
EXEC:  cpu=     220966 ela=    1194523
FETCH: cpu=          0 ela=          0
```

An Example

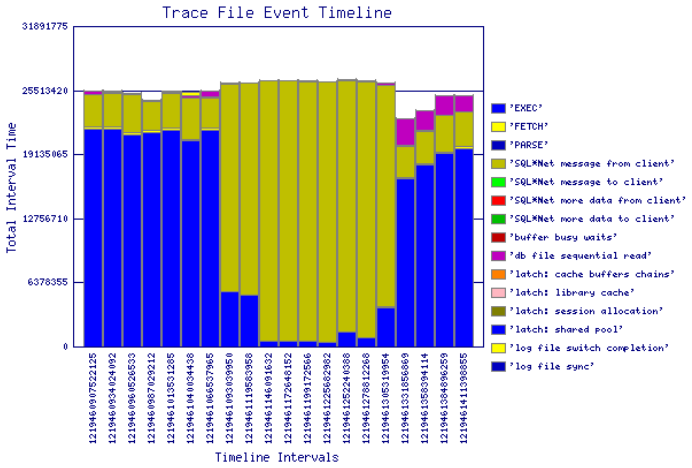
- According to the customer a process would do a lot of work, virtually stop, and then begin doing work again.
- Lots of conjecture about possible database saturation with I/O, CPU, locking and pretty much anything else folks could think of.
- A partial trace, this was a very long running process, showed a resource profile divided almost equally between "SQL*Net message from client" and "CPU Service" (i.e. CPU time) with a small amount of miscellaneous time.

Example Resource Profile

Event Name	Total Event Time (sec)	% Event Time	Total Events	Cumulative Avg Time(sec)	Cumulative Min Time(sec)	Cumulative Max Time(sec)
SQL*Net message from client	255.000799	48.11	152,647	0.001671	0.000138	0.290118
CPU Service	244.962734	46.22	864,046	0.000284	0.000000	0.054992
unaccounted-for	18.943126	3.57				
db file sequential read	10.345820	1.95	3,262	0.003172	0.000017	0.295407
SQL*Net message to client	0.364421	0.07	152,647	0.000002	0.000000	0.002573
log file switch completion	0.285168	0.05	4	0.071292	0.003946	0.239785
log file sync	0.130075	0.02	28	0.004646	0.000005	0.017599
latch: library cache	0.002779	0.00	36	0.000077	0.000005	0.001123
latch: cache buffers chains	0.002359	0.00	8	0.000295	0.000005	0.000621
SQL*Net more data from client	0.001583	0.00	201	0.000008	0.000005	0.000051
buffer busy waits	0.000276	0.00	6	0.000046	0.000017	0.000144
latch: session allocation	0.000082	0.00	1	0.000082	0.000082	0.000082
latch: shared pool	0.000062	0.00	1	0.000062	0.000062	0.000062
SQL*Net more data to client	0.000034	0.00	1	0.000034	0.000034	0.000034
Total	530.039318	100.00	1,172,888			

Example Trace Event Timeline

The National Ignition Facility



Example Summary

The National Ignition Facility

- **The Trace Event Timeline "shows" us the behavior that the customer described.**
- **It also summarizes the same information as the Resource Profile, but in this case in a more informative format.**

References

- Metalink Note: 39817.1, Interpreting Raw SQL_TRACE and DBMS_SUPPORT.START_TRACE output
- Optimizing Oracle Performance, Cary Millsap, Jeff Holt
- Interval Resource Profiler, www.appsdba.com