
ORACLE®

Common Anti-Patterns for Optimizing Big Data Oracle Databases

Vlado Barun
Real World Performance Team
March 19th, 2015

ORACLE

Safe Harbor Statement

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.

Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

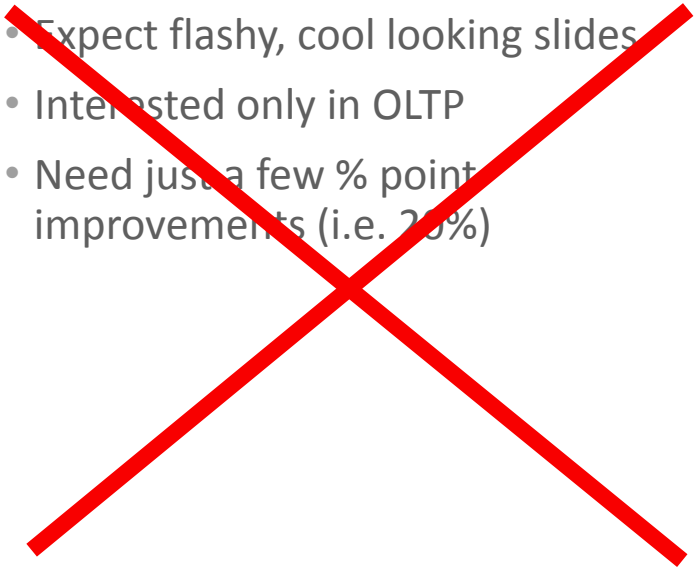
Anti-Patterns

An anti-pattern (or antipattern) is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

— Wikipedia

Is this session for YOU?

- Expect flashy, cool looking slides
- Interested only in OLTP
- Need just a few % point improvements (i.e. 20%)



- Design/Develop/Operate Decision support systems (i.e. Data Warehouse/Mart, Big Data,…)
- Require **orders of magnitude** (10x, 100x, 1000x) improvements
- Prefer an Interactive Session
- Reproduce in your own environment

Agenda

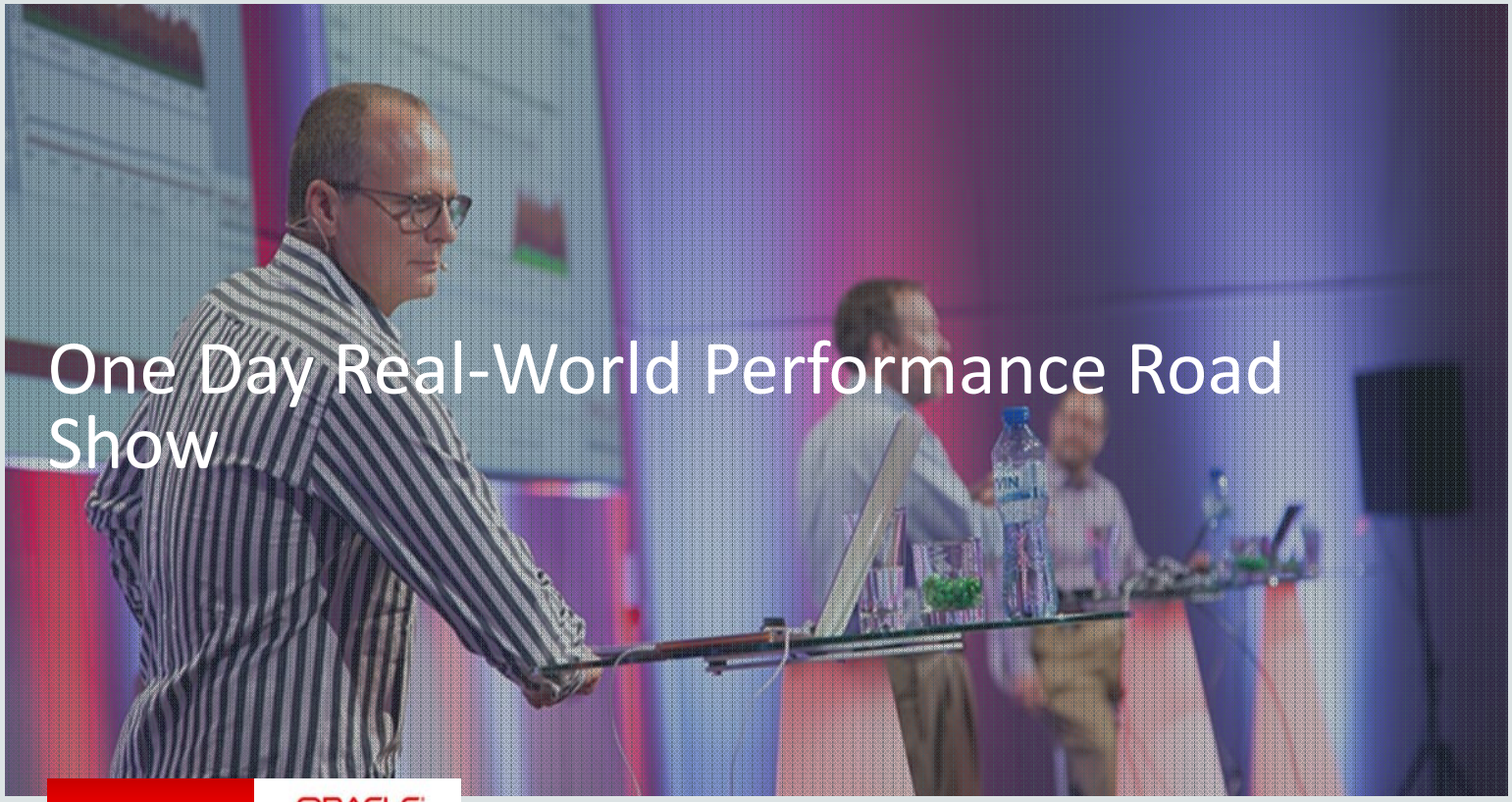
1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

Real-World Performance

Who We Are

- Part of the Database Development Organization
- Global Team located in USA, Europe, Asia
- 300+ combined years of Oracle database experience
- Innovate to achieve exceptional Database Performance
- Our methods:
  - Use the product as it was designed to be used
  - Numerical and logical debugging techniques
  - Educate others about the best performance methods and techniques
  - Avoid and eliminate “tuning” by hacking/guessing/luck

One Day Real-World Performance Road Show



ORACLE

ORACLE REAL-WORLD PERFORMANCE

Real-World Performance Tour in Association with IOUG

Featuring Andrew Holdsworth, Tom Kyte and Graham Wood

- Europe 2015
  - Feb 13th Norway
  - Mar 17th Slovenia/Croatia
  - Mar 19th Great Britain
  - Mar 23rd Denmark
  - Mar 25th Austria
  - Mar 27th Turkey
- USA—TBD
- ASIA
  - May 20th Beijing
  - May 22th Shangai
- South America—TBD

Real-World Performance

Online Video Series

- Real-World Performance Engineers discussing and demonstrating performance issues, root causes and when to apply the correct techniques
  - The Optimizer
  - Core DB Performance
  - Extreme OLTP
  - Extreme DW
- http://www.oracle.com/goto/oll/rwp

Real-World Performance Training

ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Real-World Performance Classroom Training

Classroom Training

- 4 Day Class of Intensive Performance Training
  - Topics: The Optimizer, Core DB Performance, Extreme OLTP and DW
  - Classroom, Demos, Hands On, Test and Quizzes
  - Training given by Real-World Performance Engineers
  - Designed for Architects, Developers and DBAs
  - 4 months training in 4 days
- Contact RWP or your local Oracle team to apply

Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

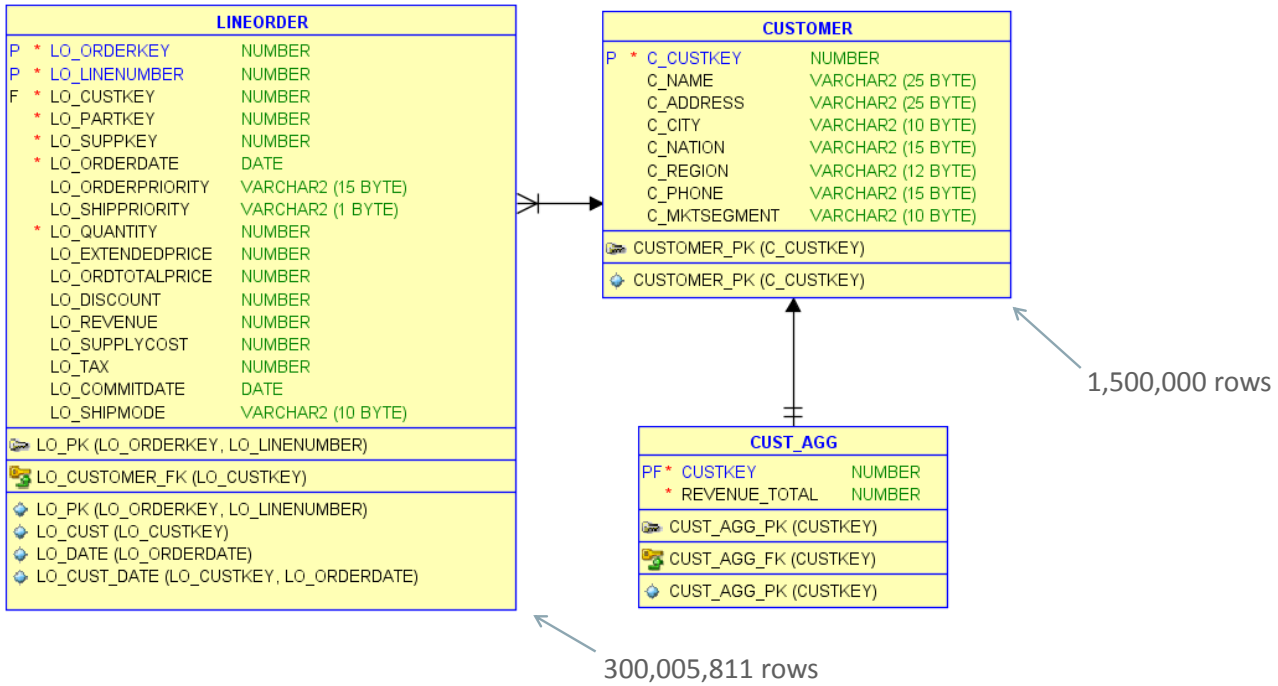
Your mission, should you choose to accept it …

- Current environment does not meet SLAs
- Business expects data volume to increase drastically in the future
- Legacy application and DB migrated to new powerful hardware but performance/scalability goals have not been reached
- Business expects 1000x better performance for a mission-critical BI process
- Your mission, should you choose to accept it, is to meet the 1000x goal

BI process details

- Identifies the most valuable customers for each month
- Currently the process takes about 26 minutes
- Implemented in Java, data in Oracle 12.1.0.2 on Exadata x3-8
- Uses 3 tables

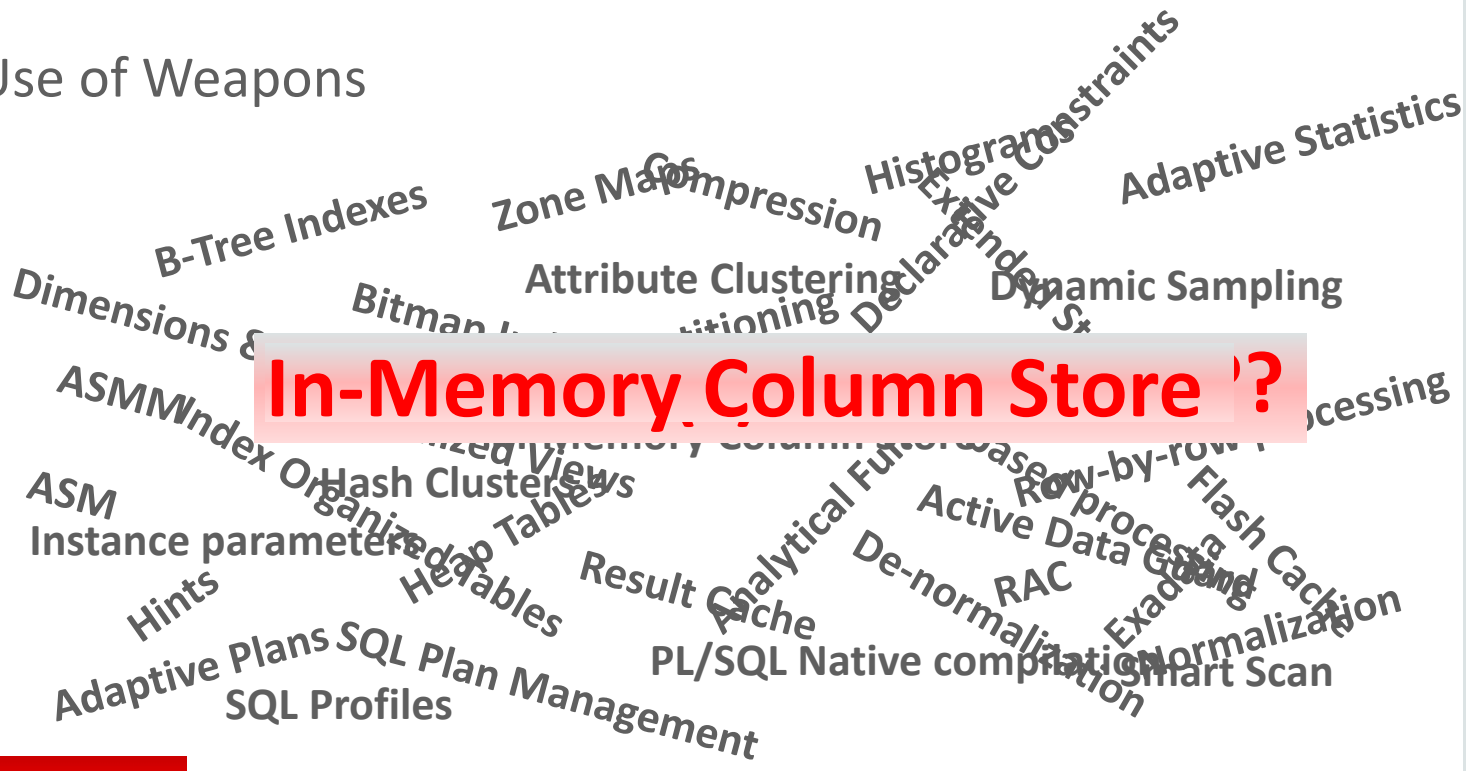
ERD



Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

Use of Weapons



Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

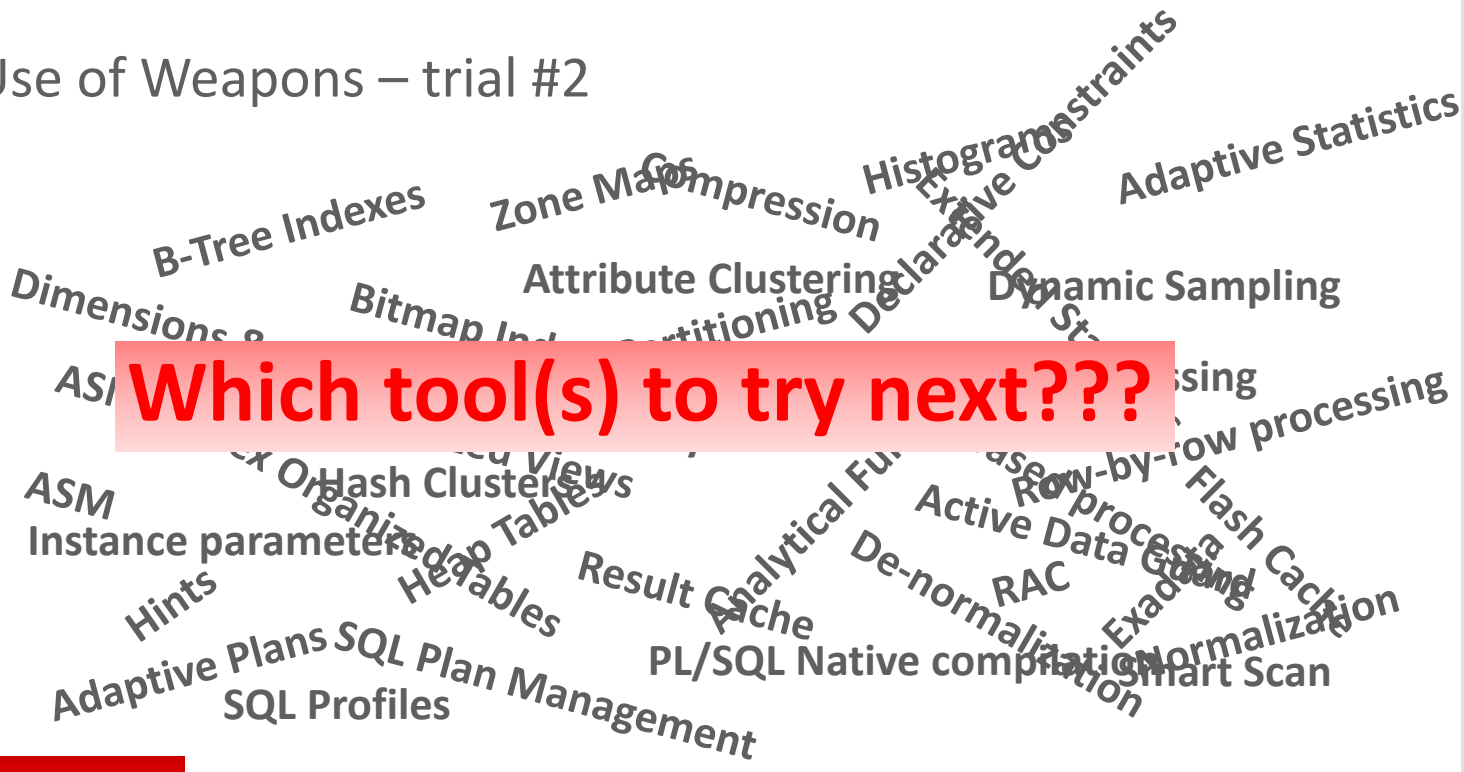
Is In-Memory the solution ?

```
SQL> Alter table lineorder inmemory memcompress for query;
Table altered.

SQL>   SELECT
2          segment_name, INST_ID
3        , populate_status
4        , round(BYTES_NOT_POPULATED /1024/1024/1024, 2) as gb_not_populated
5    FROM gv$im_segments
6    where owner='APPS1000X'
7    order by segment_name, inst_id
8  ;

SEGMENT_NAME            INST_ID POPULATE_ GB_NOT_POPULATED
-------------------- ---------- --------- ----------------
CUSTOMER                     1 COMPLETED                0
LINEORDER                    1 COMPLETED                0
```

Use of Weapons – trial #2



Anti-Pattern #1 - Trial and Error approach

- Time consuming
- Relies on luck
- Limits understanding of the root cause of a performance bottleneck
- Instead, “Look before you leap” principle

Look before you leap

The universal experience of programmers
who have been using measurement tools
has been that their intuitive guesses fail.

- Donald Knuth

Look before you Leap

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
SSB	906471671	SSB1	1	15-Feb-15 17:02	12.1.0.2.0	YES

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
slcg02db01.us.oracle.com	Linux x86 64-bit	160	80	8	2019.52

	Snap Id	Snap Time	Sessions	Cursors/Session	Instances
Begin Snap:	3724	16-Feb-15 13:14:25	135	1.2	1
End Snap:	3725	16-Feb-15 13:40:26	135	1.2	1
Elapsed:		26.02 (mins)			
DB Time:		13.42 (mins)			

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.5	9.7	0.00	0.00
DB CPU(s):	0.5	9.7	0.00	0.00

Look before you Leap

- Notice elapsed time >> db time
- Notice db cpu >> sql execution elapsed time
- Notice sum of all sql times << db cpu
- We can only account for ***34%*** of the total elapsed time

Where is the missing time?!

- java process OS stats

```
$ time java demo
 4.608u 2.780s 26:08.47 0.4%
```

- If 50% time is not spend in the App nor in the database, where is it?
- What is between the App and the DB?
- 3.3 M SQL*Net roundtrips to/from client = 2,109/sec

Anti-Pattern #2 – Stay inside your Comfort Zone

- Don’t choose your tool solely based on what you are comfortable with, or what is currently in vogue, …
- Choose the tool that can achieve the goal
- When processing large amounts of data, move the processing close to the source of the data

Eliminate the bottleneck – network round-trips

One way to eliminate network round-trips is to move the data processing logic into the database, i.e. using PL/SQL.

```
SQL> exec find_high_value_customers
```

Eliminate the bottleneck - network round-trips

```
SQL>create or replace procedure find_high_value_customers as
  2      revenue_total cust_agg.revenue_total%type;
  3  begin
  4
  5    execute immediate 'truncate table cust_agg';
  6
  7    for c in (select c_custkey, c_name  from customer)
  8    loop
  9        revenue_total := 0;
 10
 11        for lo in ( select lo_revenue
 12                      from lineorder
 13                     where lo_orderdate between date'1996-12-01' and date'1996-12-31'
 14                       and lo_custkey = c.c_custkey)
 15        loop
 16            revenue_total := revenue_total + lo.lo_revenue;
 17        end loop;
 18
 19        insert into cust_agg (custkey, revenue_total) values (c.c_custkey,revenue_total);
 20    end loop;
 21
 22    delete cust_agg where revenue_total < 200000000;
 23    commit;
 24
 25  end;
 26  /

Procedure created.

Elapsed: 00:00:00.07
```

Look before you Leap – PL/SQL

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
SSB	906471671	SSB1	1	15-Feb-15 17:02	12.1.0.2.0	YES

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
slcg02db01.us.oracle.com	Linux x86 64-bit	160	80	8	2019.52

	Snap Id	Snap Time	Sessions	Cursors/Session	Instances
Begin Snap:	3728	16-Feb-15 14:45:56	134	1.2	1
End Snap:	3729	16-Feb-15 14:51:39	135	1.2	1
Elapsed:		5.71 (mins)			
DB Time:		5.72 (mins)			

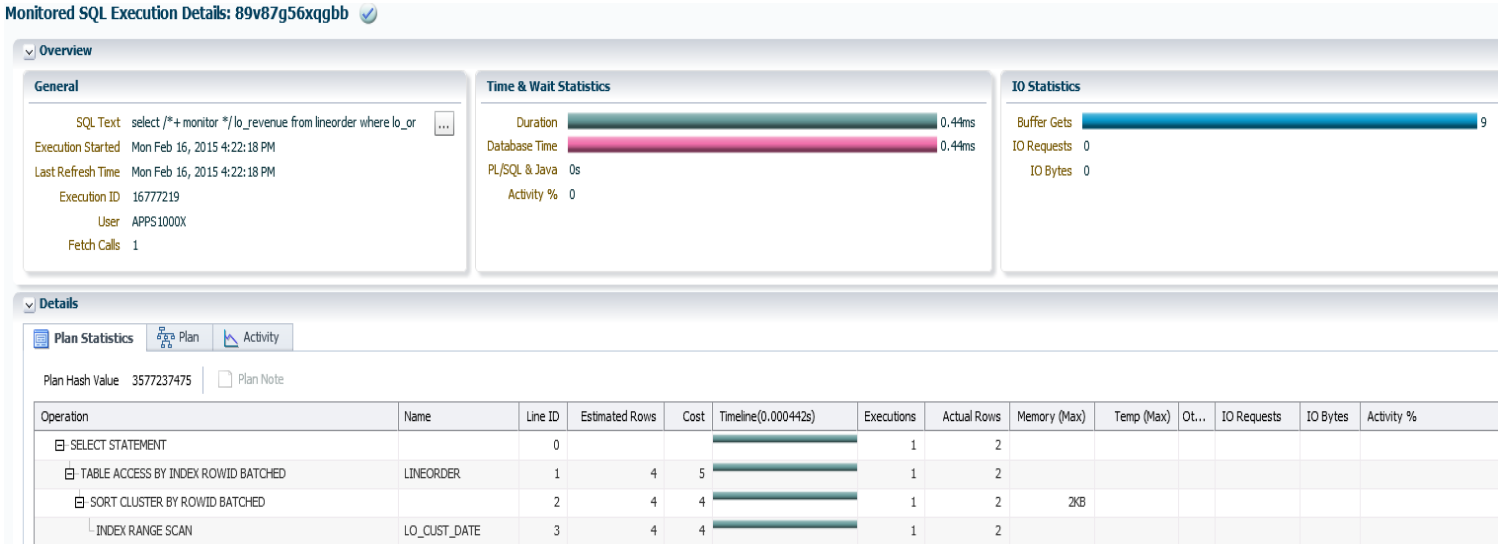
Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.0	9.3	0.00	0.63
DB CPU(s):	1.0	9.2	0.00	0.63

Why did the performance improve?

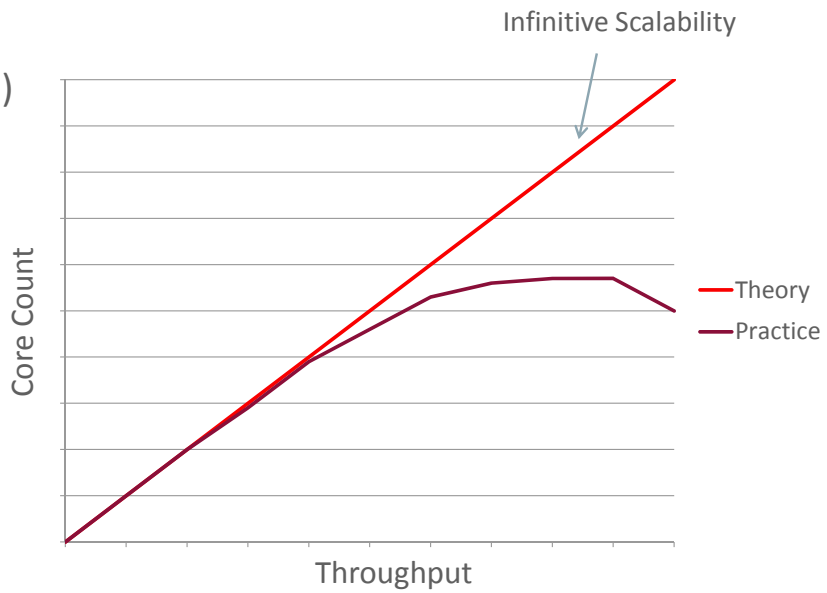
- AWR
  - elapsed time ≈ db time
  - db cpu ≈ sql execution elapsed time
  - sum of all sql times ≈ db cpu
  - Pl/sql time ≈ 21% of baseline + ≈ 70% of missing time (network round trips)
  - SQL*Net roundtrips to/from client reduced from about 3,300,000 to 287
  - Network traffic reduced from 514 MB to 0.3MB (330KB)
  - Not about Java vs PL/SQL, or App vs DB
  - Use the tool that can achieve the goal
  - Approx. 5x improvement

Can we tune the query?



Anti-Pattern #3 – Throw Hardware at it

- Use Faster CPUs?
  - Intel® Xeon® E7-8870 (2.40 GHz)
  - 384 GFLOPs
- More CPUs



Identify the bottleneck

- What is the alternative?
- Can the goal be accomplished using a different approach/algorithm?

Anti-Pattern #4 – Looking only at the tree…

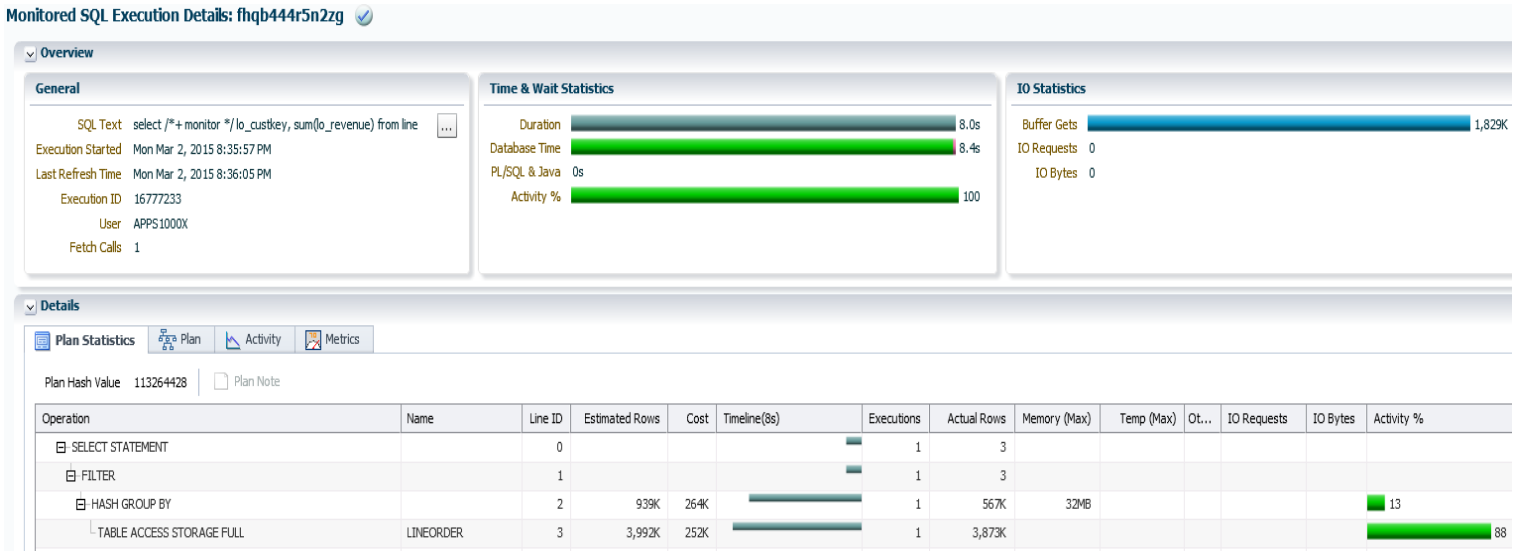
- Miss the forest for the trees
- Row-by-row processing = Slow-by-slow processing
- How to use a Tesla Model S P85D to pickup groceries
- Applying OLTP specific algorithms to Big Data Systems
  - Another example of Anti-Pattern #2 – Stay inside your Comfort Zone

Eliminate the bottleneck – row-by-row processing

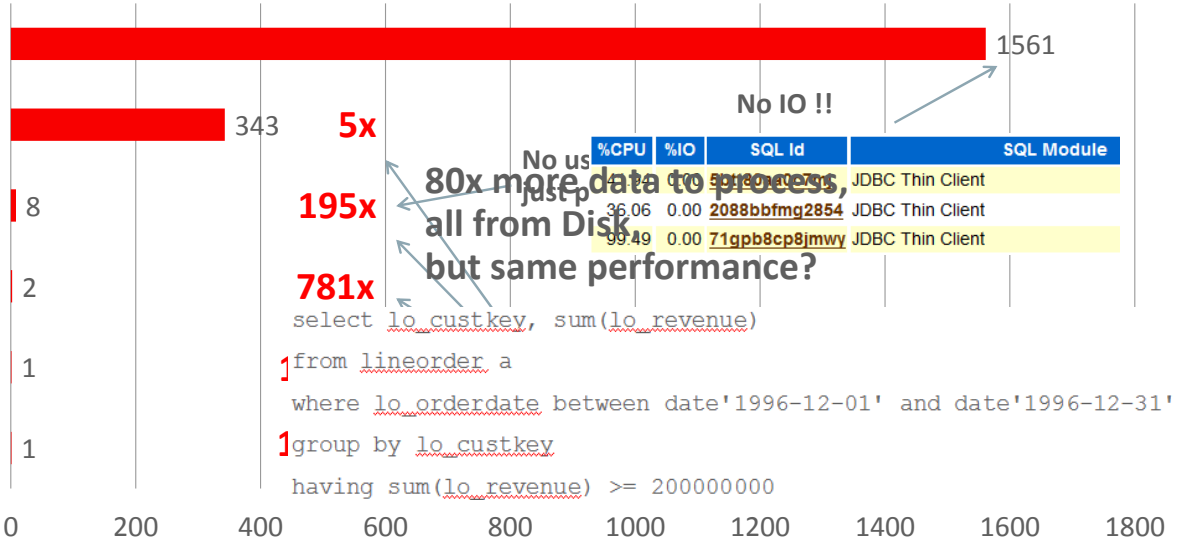
- Replace row-by-row with set-based processing
- Just tell the database what data you want, not how to get to it !

```
select lo_custkey, sum(lo_revenue)
from lineorder a
where lo_orderdate between date'1996-12-01' and date'1996-12-31'
group by lo_custkey
having sum(lo_revenue) >= 200000000
;
```

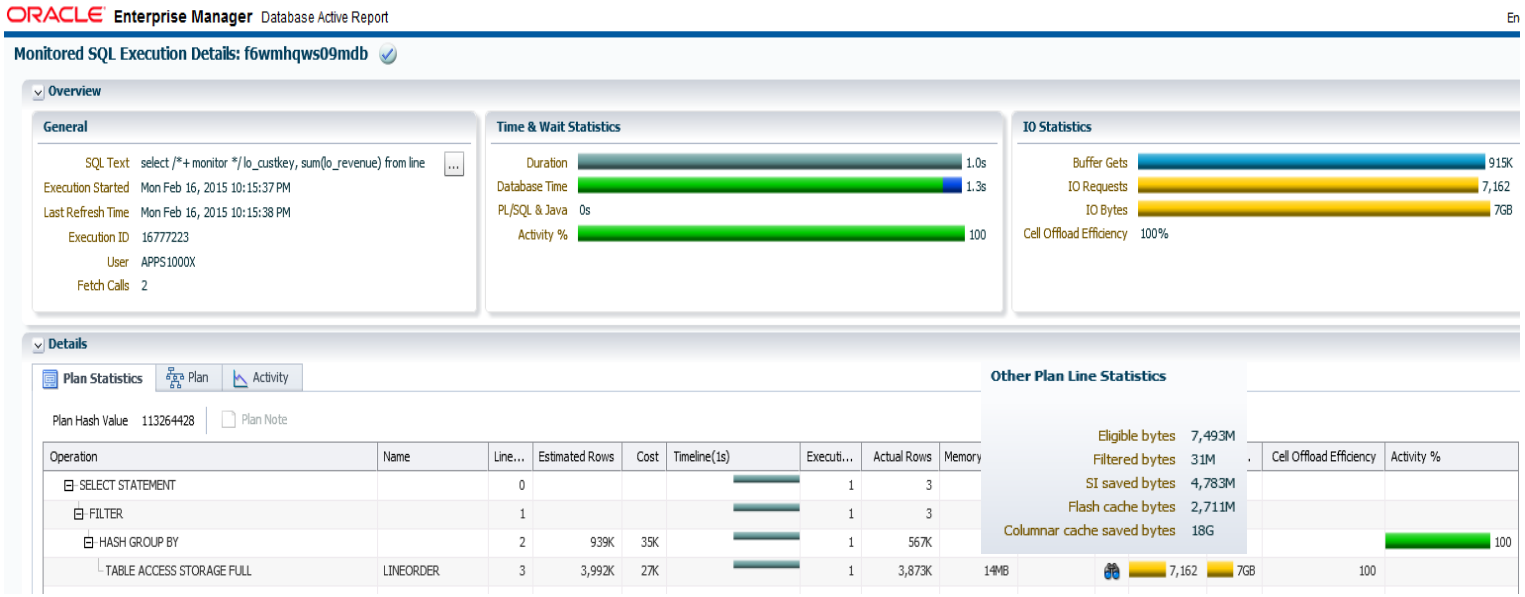
Set based processing



Results Comparison (Elapsed seconds)



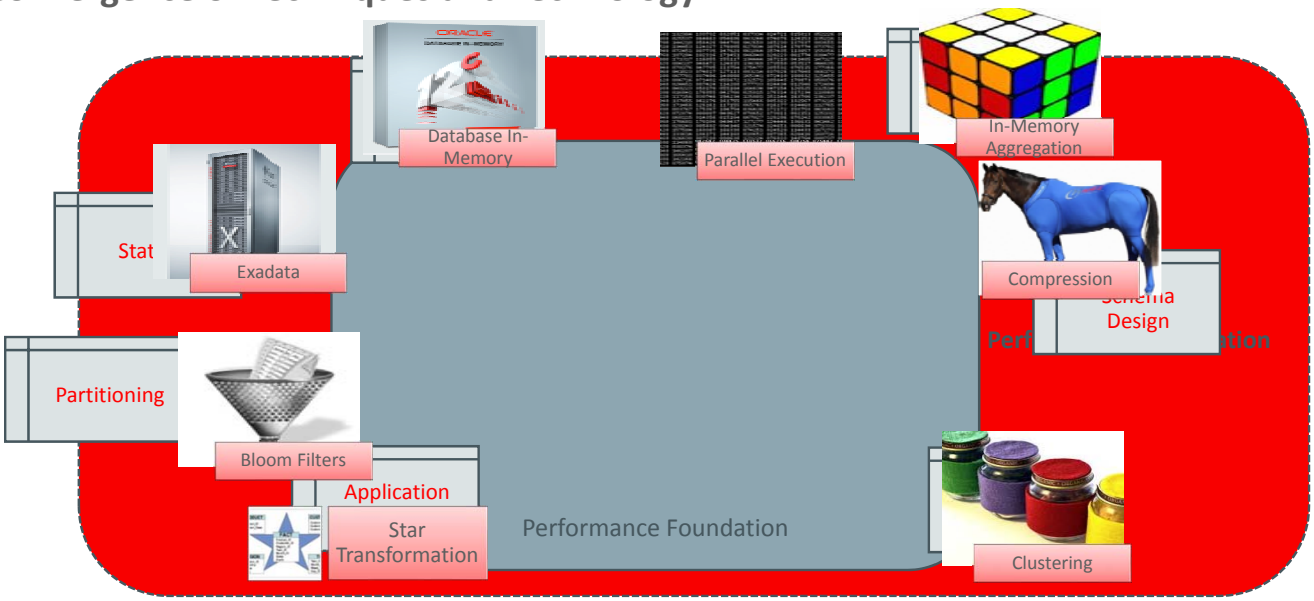
Set based processing (IO & Exadata)



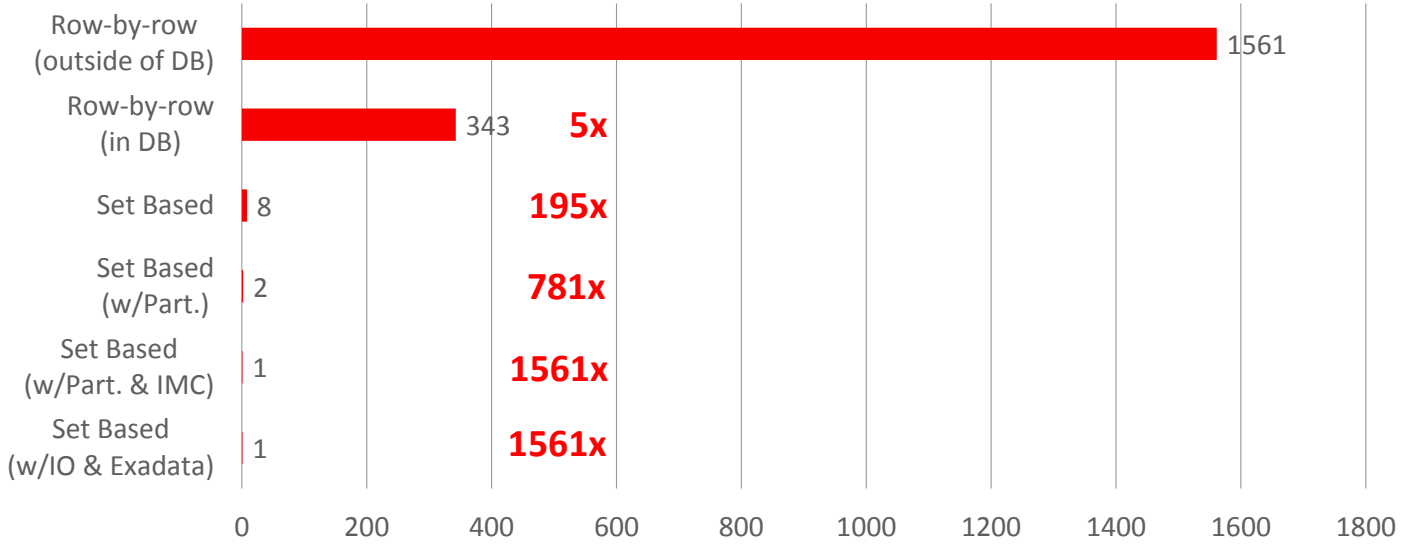
Use of Weapons

Real World Performance

Convergence of Techniques and Technology



Where is the biggest bang for the buck (ROI)?



Agenda

1. Is this session for YOU?
2. About the Real World Performance Team
3. Your mission, should you choose to accept it …
4. Use of Weapons
5. Solution (or How to use a Model S P85D to pickup groceries)
6. 3 takeaways for achieving Extreme Performance

1. Look before you leap

It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

— Sherlock Holmes,

"The Adventures of Sherlock Holmes"

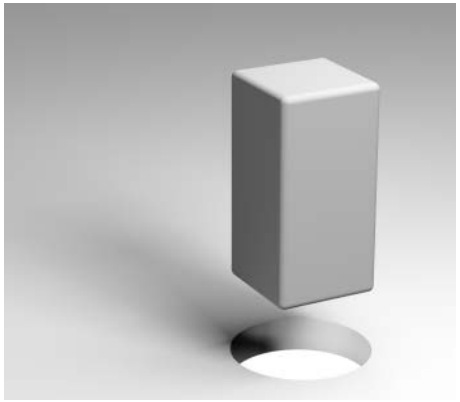
by Sir Arthur Conan Doyle

The truth is out there*

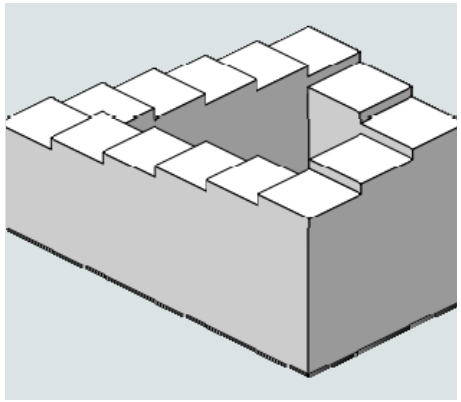
- EM
- EM Express Performance Hub
- AWR
- ADDM
- ASH
- SQL Monitor

- SQL Trace
- SQL Compiler Trace
- SQL Test Case Builder
- SQLTXPLAN
- SQLD360
- etc.

2. Root Causes of Suboptimal Database Performance



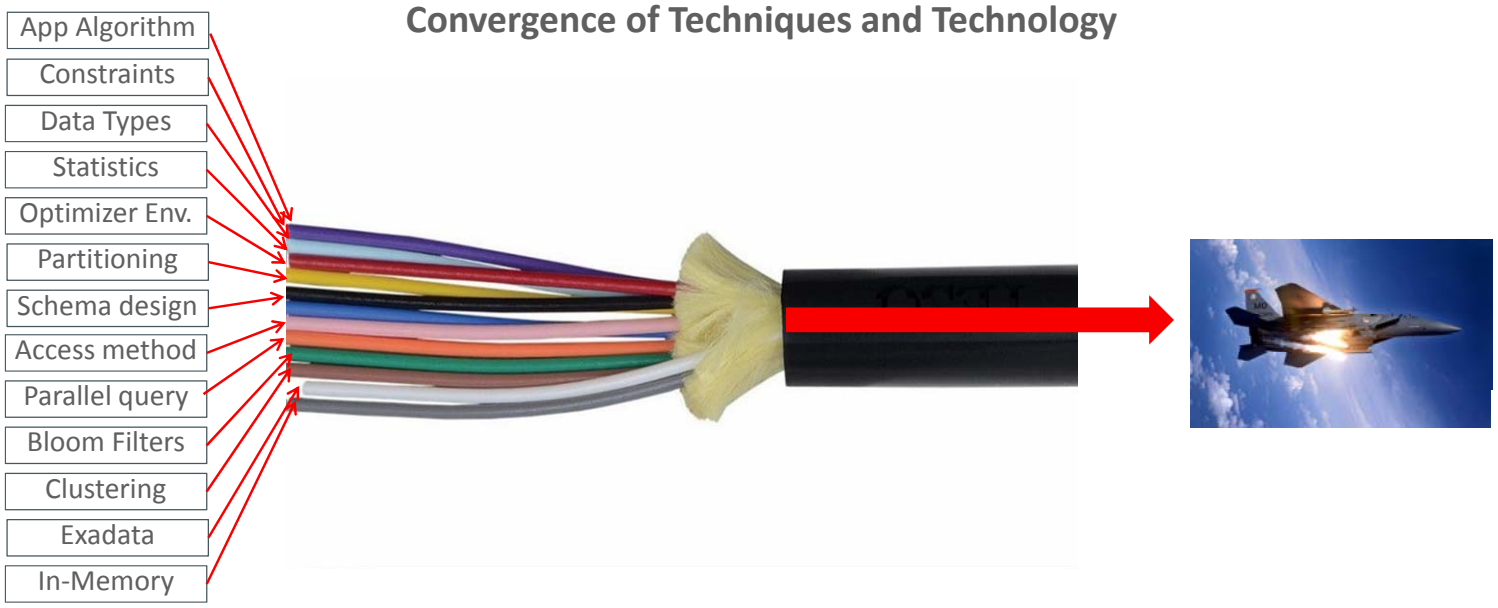
The database is not being used as it was designed to be used



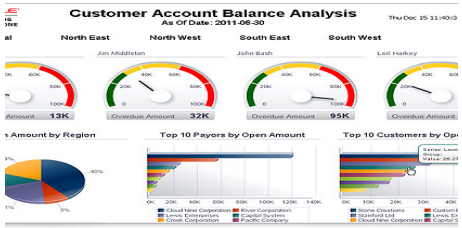
The application architecture/code design is suboptimal

There is a suboptimal algorithm in the database

3. Extreme Performance is multi-faceted



Recent Results 1000X Projects



Baseline:	~ 4.3 Hours
Code Changes:	4.3 Hours
Correct Usage:	29 secs
Bug Fixes:	11.5 secs
Final:	11.5 secs
Speed up:	**1346x**

Baseline:	~ 2.4 Days
Code Changes:	27 Mins
Correct Usage:	7.5 Mins
Bug Fixes:	3 Mins 27 Secs
Final:	3 Mins 27 Secs
Speed up:	**1002x**

Baseline:	4.06 hours
Code Changes:	3.65 Secs
Correct Usage:	3.65 Secs
Bug Fixes:	3.65 Secs
Final:	3.65 Secs
Speed up:	**4007x**

Environment Setup

- Create a new*1 or use an existing DB
- Create objects using DDL provided in the Appendix
- Generate Data using dbgen*2
- Import data (i.e. using external tables)
- Have fun

*1 i.e. http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html

*2 i.e. download from https://github.com/electrum/ssb-dbgen

Hardware and Software
Engineered to Work Together


ORACLE