CHEP 2007

Building a Scalable Event Level Metadata Service for ATLAS

Performance and Scalability Tests for
Relational TAG Database

Jack Cranshaw, Luc Goossens, David Malon,
***Helen McGlone*, Florbela Viegas**

Introduction

- The ATLAS TAG Database
- A Challenging Environment
- The 1TB Tests
- Learning
- Performance results
- What's next

The ATLAS TAG Database

- Defined in ATLAS Computing Model
- File and relational database system
- Support seamless discovery, identification, selection and retrieval of events of interest to analysis and no others
- TAGs written at AOD production
- Implemented using LCG POOL Collection infrastructure

What is a TAG?

- 1kB summary information per event
- Attributes selected to support selection of events and navigation within the system
  - Event ID and Global event properties
  - Trigger information
  - Quality information
  - Temporal information
  - Physics objects (high level)

A Challenging Environment.....

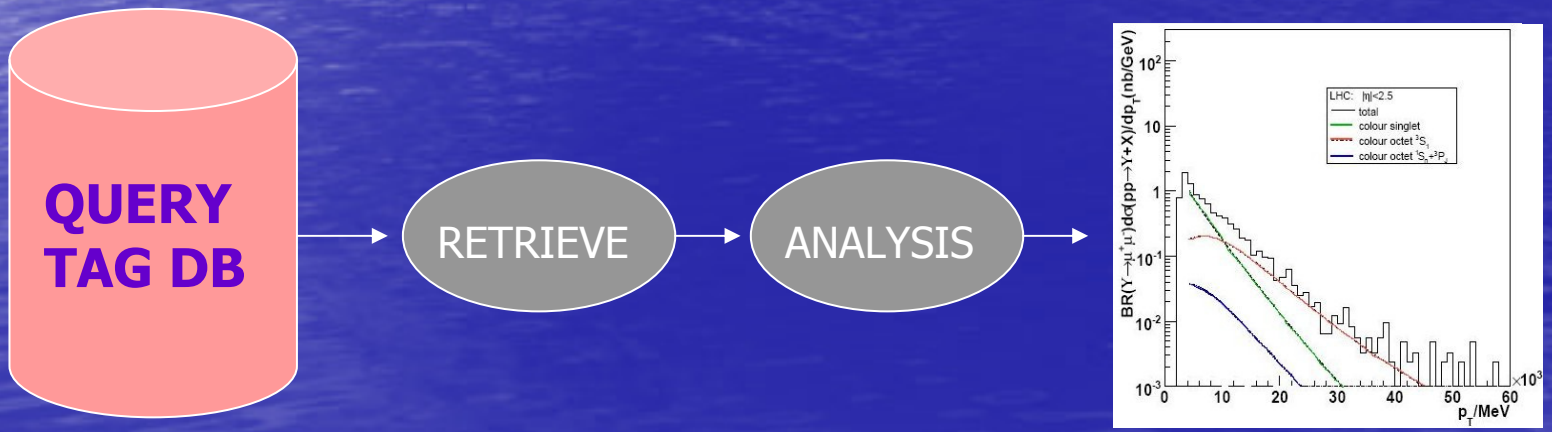
- ATLAS data rate – 200Hz
  - 200 new event TAGs per second for data taking, assume 50K active seconds/day = 58% efficiency for each active day, 10^7 events/day
- ATLAS data volume

Year	% Year data taking	Data Volume
2008	40	1.42TB
2009	60	3.65TB
additional	60	6.09TB

Terabytes!

A Challenging User!

- The ATLAS physicist…..
  - Fast, efficient, accurate queries
  - Reliable navigation to event data
  - Seamless integration with analysis



Performance and Scalability Tests for Relational TAG Database

- Large scale realistic tests to uncover challenges brought with scale
- Optimise and measure performance
  - Management
  - Partitioning
  - Indexing
  - Optimizer
  - Hints
  - Multiple clients
  - Query patterns

Why 1TB tests?

- We expect that important performance and management phase transitions will be crossed as we scale to billions of events
- Queries may be unselective and select across a range of attributes – want to address this challenge at large scale
- Memory to disk

The 1TB TAG Database

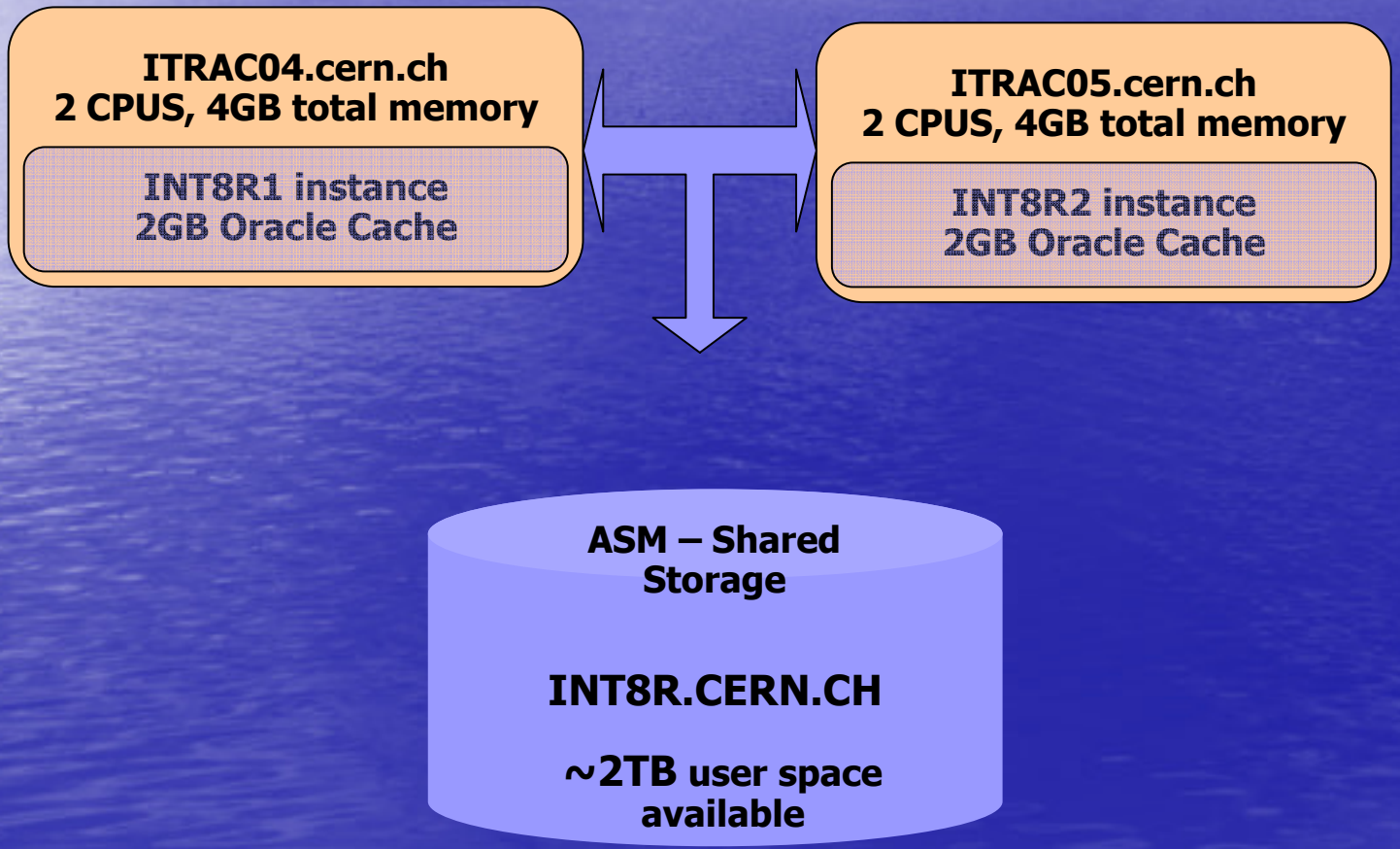
- Created one million dummy events based on real TAG attributes
- Realistic and varied data types and value distributions
- Multiplication and replication, one million to one billion, realistic and with unique ids

The 1TB Table

Name	Datatype	Size	Scale	Nulls?	Index ?
ID	NUMBER	36	0	Yes	B*Tree
RUNNR	NUMBER	12	0	Yes	Bitmap
EVENTNR	NUMBER	12	0	Yes	
GOLDEN1	NUMBER	12	0	Yes	Bitmap
GOLDEN2	NUMBER	12	0	Yes	
AODFILEFK	NUMBER	12	0	Yes	Bitmap
ESDFILEFK	NUMBER	12	0	Yes	
BOOL500CHAR01	CHAR	1		Yes	Bitmap
BOOL100CHAR01	CHAR	1		Yes	Bitmap
BOOL10CHAR01	CHAR	1		Yes	Bitmap
BOOL1CHAR01	CHAR	1		Yes	Bitmap
BOOL500NUM01	NUMBER	1	0	Yes	Bitmap
BOOL100NUM01	NUMBER	1	0	Yes	Bitmap
BOOL10NUM01	NUMBER	1	0	Yes	Bitmap
BOOL1NUM01	NUMBER	1	0	Yes	Bitmap
ENUMUNI100VC01	VARCHAR2	10		Yes	Bitmap
ENUMUNI10VC01	VARCHAR2	10		Yes	Bitmap
ENUMEXP1000NUM01	NUMBER	5	0	Yes	Bitmap
ENUMEXP100NUM01	NUMBER	5	0	Yes	Bitmap
ENUMEXP10NUM01	NUMBER	5	0	Yes	Bitmap
ENUMUNI1000NUM01	NUMBER	5	0	Yes	Bitmap
ENUMUNI100NUM01	NUMBER	5	0	Yes	Bitmap
ENUMUNI10NUM01	NUMBER	5	0	Yes	Bitmap
ENUMUNI5NUM01	NUMBER	5	0	Yes	Bitmap
NOR100BF01	BINARY_FLOAT	4		Yes	B*Tree
NOR10BF01	BINARY_FLOAT	4		Yes	B*Tree
NOR1BF01	BINARY_FLOAT	4		Yes	B*Tree
NOR100NUM01	NUMBER	12	5	Yes	B*Tree
NOR10NUM01	NUMBER	12	5	Yes	B*Tree
NOR1NUM01	NUMBER	12	5	Yes	B*Tree
UNINUM01	NUMBER	12	0	Yes	B*Tree
UNI10KNUM01	NUMBER	12	0	Yes	Bitmap
UNI1KNUM01	NUMBER	12	0	Yes	Bitmap
UNI100NUM01	NUMBER	12	0	Yes	Bitmap
UNI10NUM01	NUMBER	12	0	Yes	Bitmap

- We replicate the 1M rows to produce a 1TB table
- Approximately 1kB row size (expected TAG size)
- Distribution of values tries to mimic the types of columns and distributions expected for TAG metadata.
- *This set of attributes is repeated in the same sequence until set 8, but only this set is indexed.*

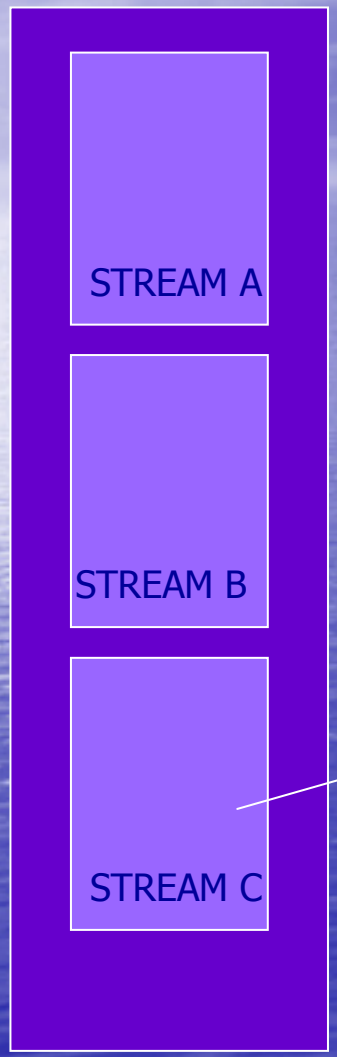
1TB Test system



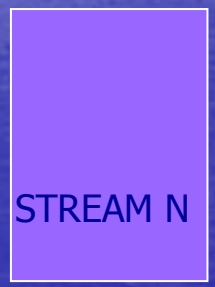
Partitioning the data

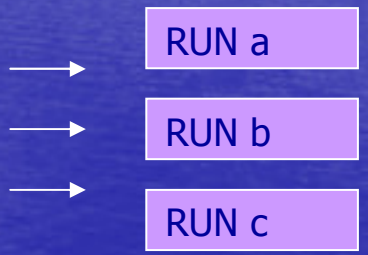
- Partitions are an Oracle tool
  - Performance queries
  - Data manageability
- Multidimensional
  - Horizontal
  - Vertical
- Choice of partition key
  - RUN
  - STREAM

Table Architecture



Divide all data HORIZONTALLY into TEN STREAM TABLES





Partition all STREAM TABLES HORIZONTALLY by RUN – each table has 100 partitions, 40 runs per partition, 1 million rows per partition

- This is our nth partitioning schema…..

Indexing

- In our test table we have btree index, bitmap index and non indexed attributes
- Tested extensively to understand performance and management implications of all three
- We learned
  - Btree and bitmap index attributes perform differently
  - Oracle has distinct query plans which are optimal for each
  - Non indexed attributes are a huge restriction on query performance
- So…..
  - Index everything (challenge)
  - Pre-process to separate btrees and bitmaps, implement a different query plan for each

Optimizing the Optimizer

- Oracle has an Optimizer designed to select optimal query execution plans
- Performance is sensitive to query plans
  - Optimizer chooses the plan based on cost estimation
  - Complex selection
  - Uses statistical information about data
- Optimizer is not perfect
  - Investigate optimal query plans
  - Help the Optimizer make the right choice

Oracle Hints

- Many hints were tested…..

PARALLEL - needed for parallelization of full table scan or partition range scan
PARALLEL INDEX - needed for parallelization of index access
INDEX JOIN - for hash joins with b-tree indexes
INDEX COMBINE - for bitmap indexes
opt param(INDEX JOIN ENABLED ,false) - Can enable and disable session parameters for a single SQL

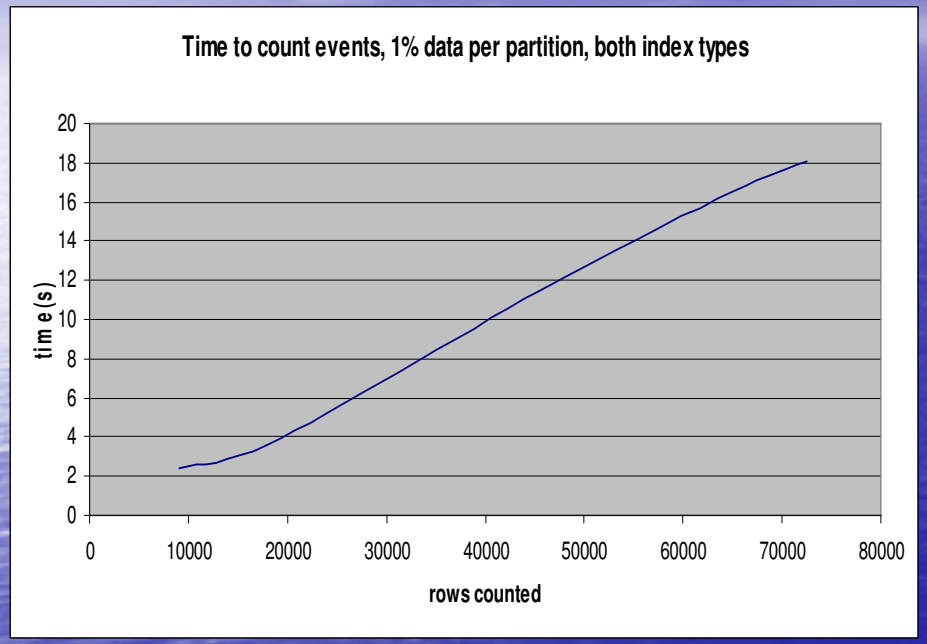
Queries for testing

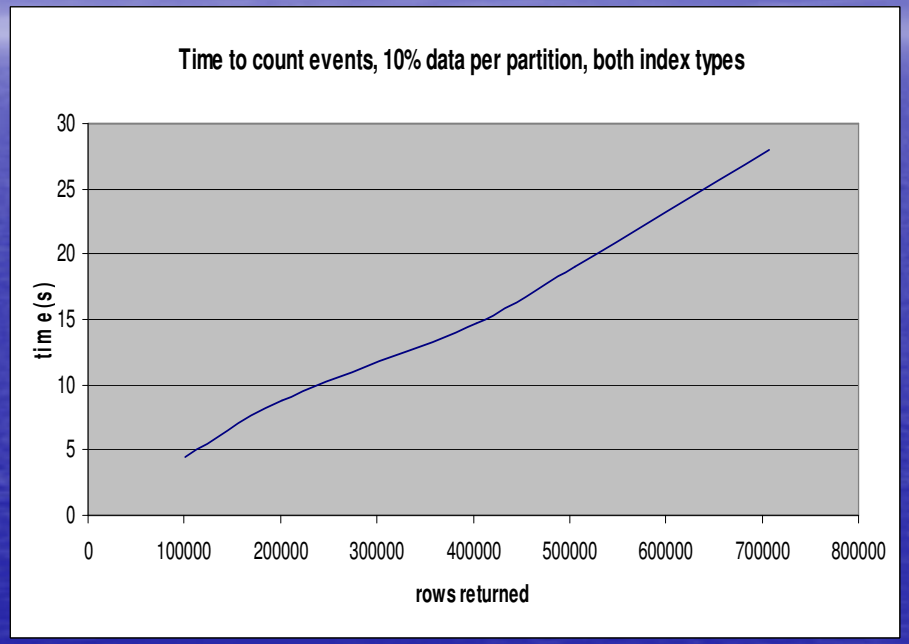
Count the events with at least two electrons and missing ET >10GeV that are "good for physics" - **SUMMARY**

Give me all the events with at least two electrons and missing ET >10GeV that are "good for physics" - **CONTENT**

- Selection based on both index types
- Use INDEX_JOIN for btrees and INDEX_COMBINE for bitmaps, then INTERSECT
- Flush buffer cache between queries, so no cache advantage
- Increase number of partitions involved as increase rows returned, consistent % rows from each partition

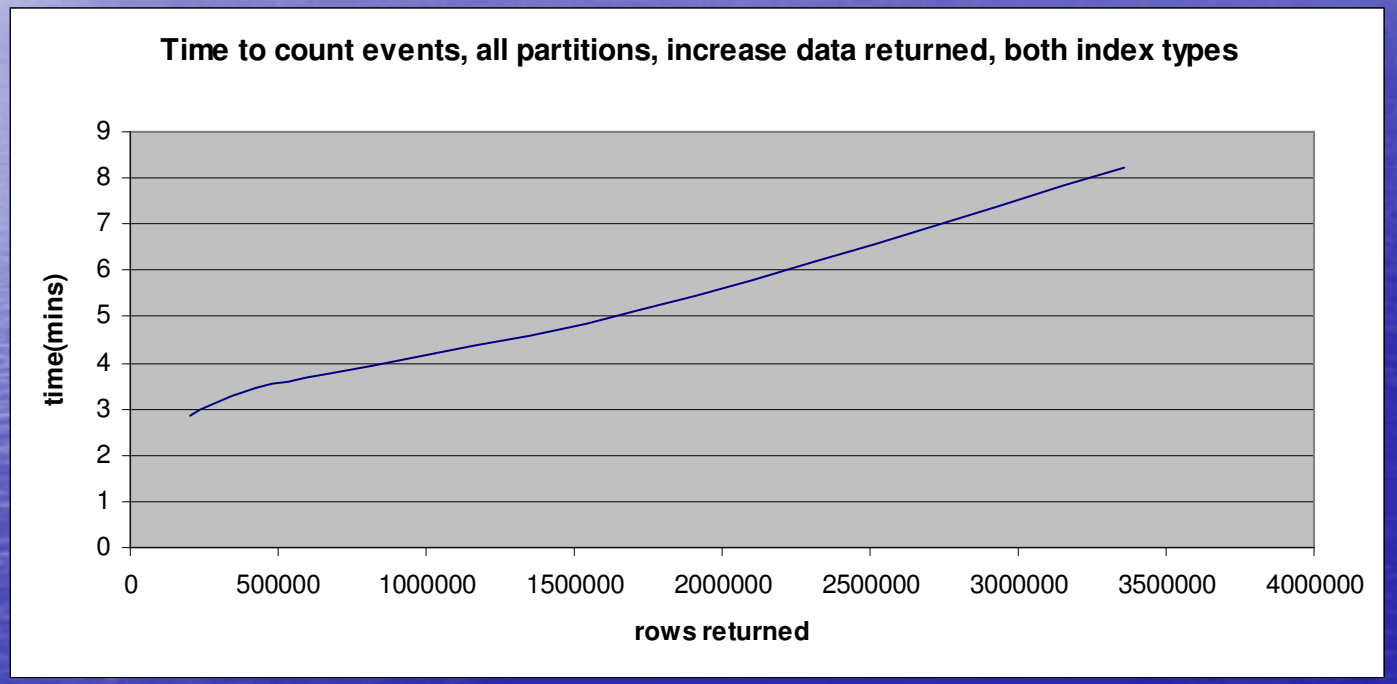
Summary queries





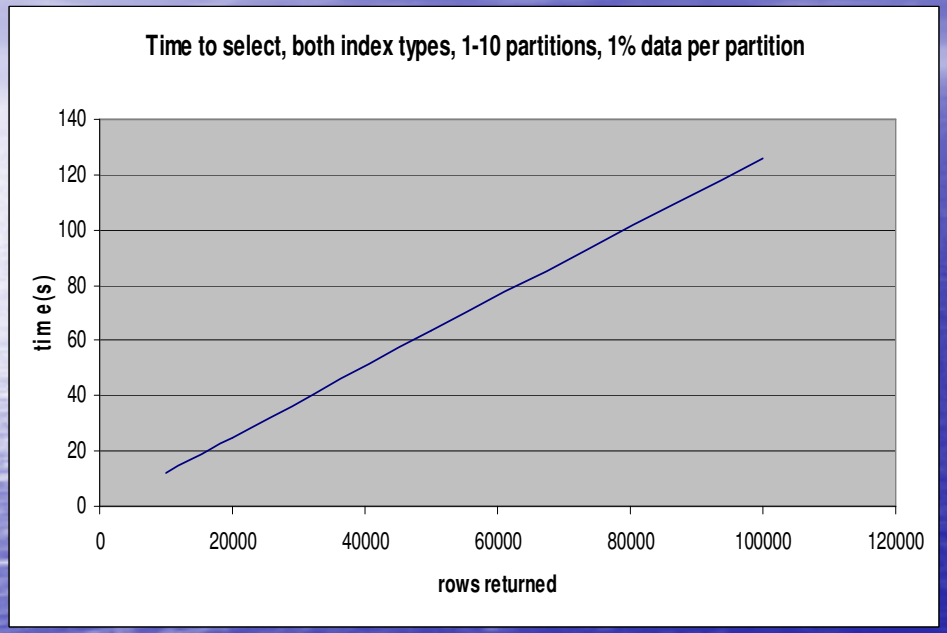
- Time increases with number of partitions
- Linear increase – predict time
- 10 x data not 10 x time – can predict time with some bounds
- Time in order of seconds

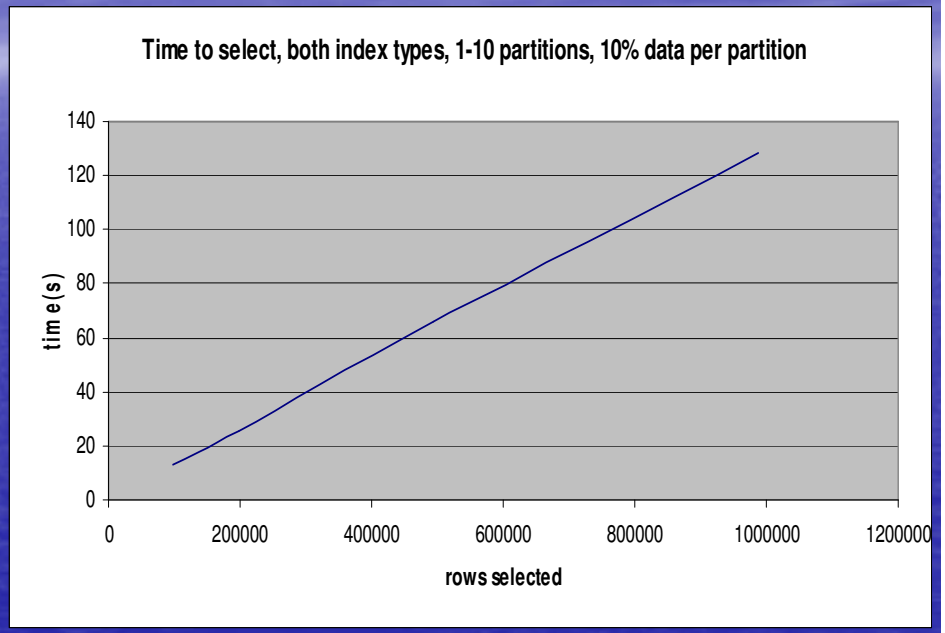
Summary queries – an extreme



See value of partition elimination

Content queries





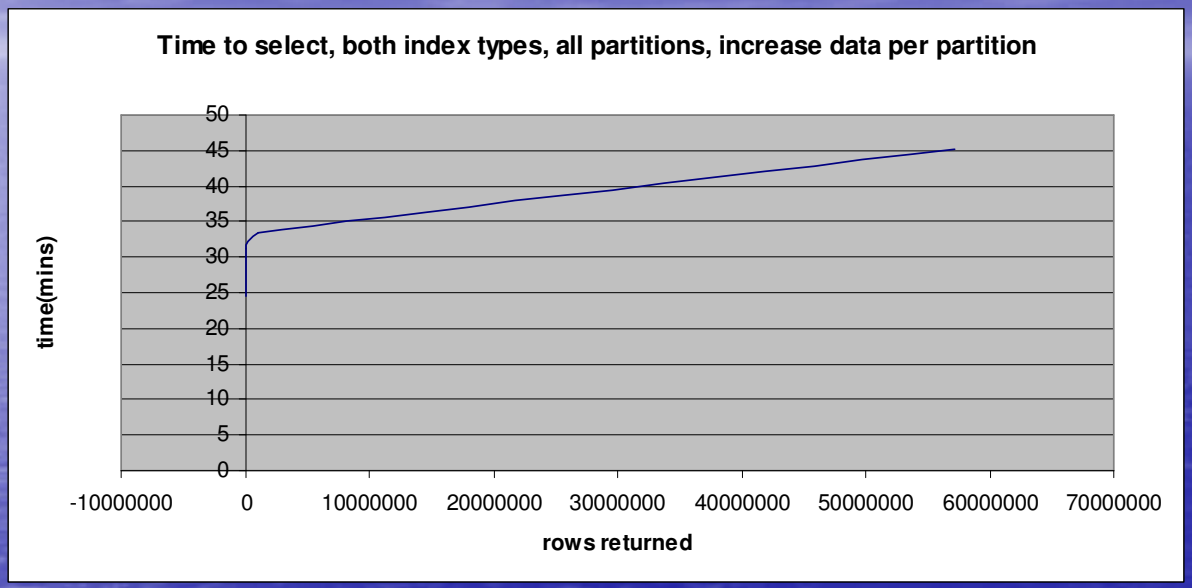
Linear increase in time with number of partitions

Time overhead is in number of partitions accessed, not data returned from within

Time order of seconds

Does this linear relation extrapolate indefinitely…..

Content query – an extreme

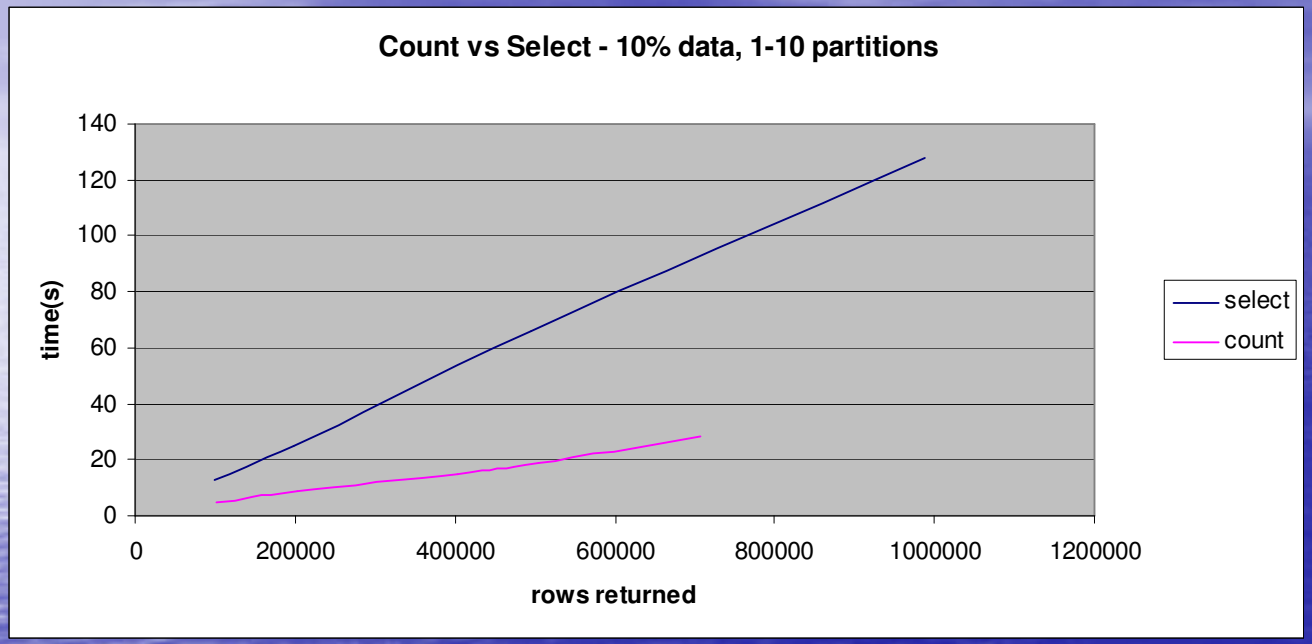


If linear relation is constant ~proportional to number of partitions in query, then query from all (100) partitions ~20 mins
Not in practice….why?

Threshold case where sorts move from all memory to disk – higher performance overhead, same query plan, but with use of disk

Faster than full table scan

Results

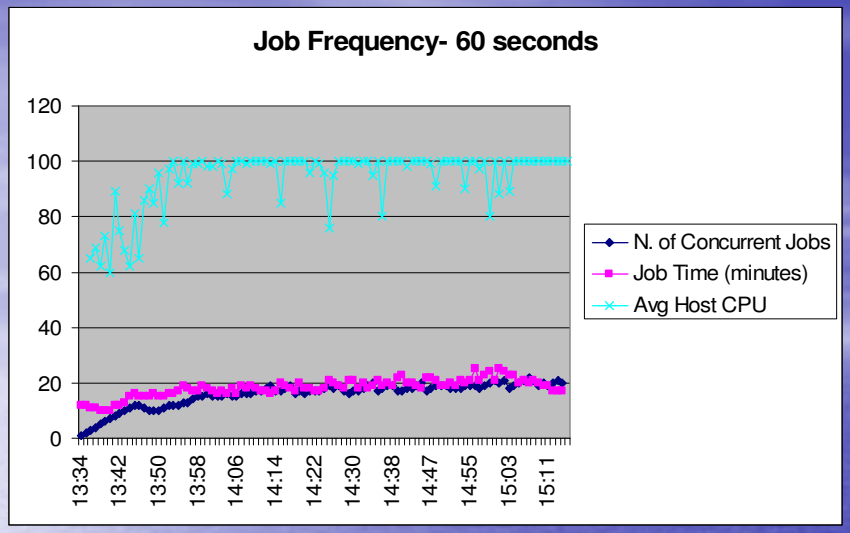


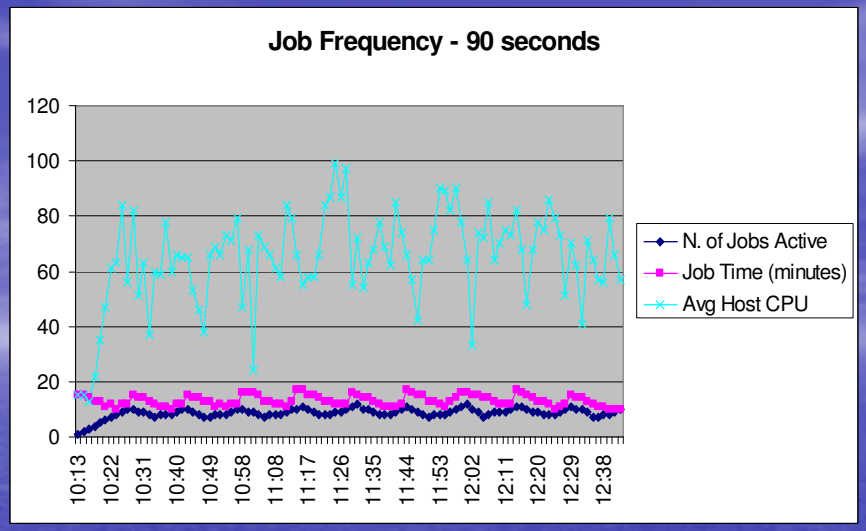
The "extreme case" (optimised) - upwards of 30 mins for selects

- Encourage counts before select
- Encourage use of partition key
- Can extrapolate to predict time query likely to take
- Pre-processing crucial

Multiple clients – Stress Tests

- Assess performance of system under multi client environment
- Simulate a realistic query environment by creating expected typical queries
  - Counts and retrieves
  - Vary attributes and filters
- Create a sample job of 9 optimised queries, with 60 seconds, then 90 seconds of pause between each
- Sessions on one node of INT8R cluster.
- Node has 2Gb memory and 2 CPUs





- Saturation at 60 seconds, queries supported at 90 seconds
- 1 Job every 90 seconds generates ~9000 queries per day
- T0 production database has 6 nodes, each T1 has 2 nodes (3 oracle licences)
- New hardware in April 2008, will test when new hardware becomes available
- Need to limit concurrent processes

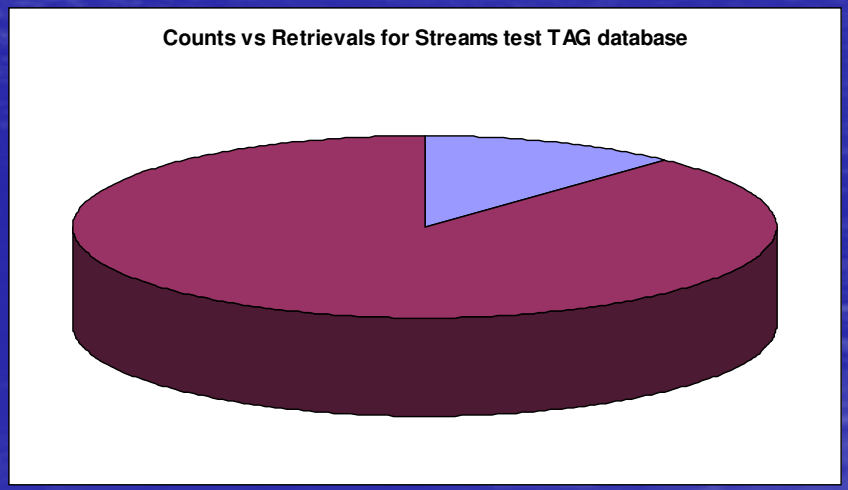
ATLAS Relational TAG Database experience

- The Streams Test TAG database
  - Smaller scale
  - Introduce users to TAGS
  - Gather query pattern information

http://atlas.web.cern.ch/Atlas/GROUPS/OPERATIONS/dataBases/TAGS/tag_browser.php

> 5000 queries captured

"Pinch of salt" for query capture….. but positive and useful feedback nonetheless

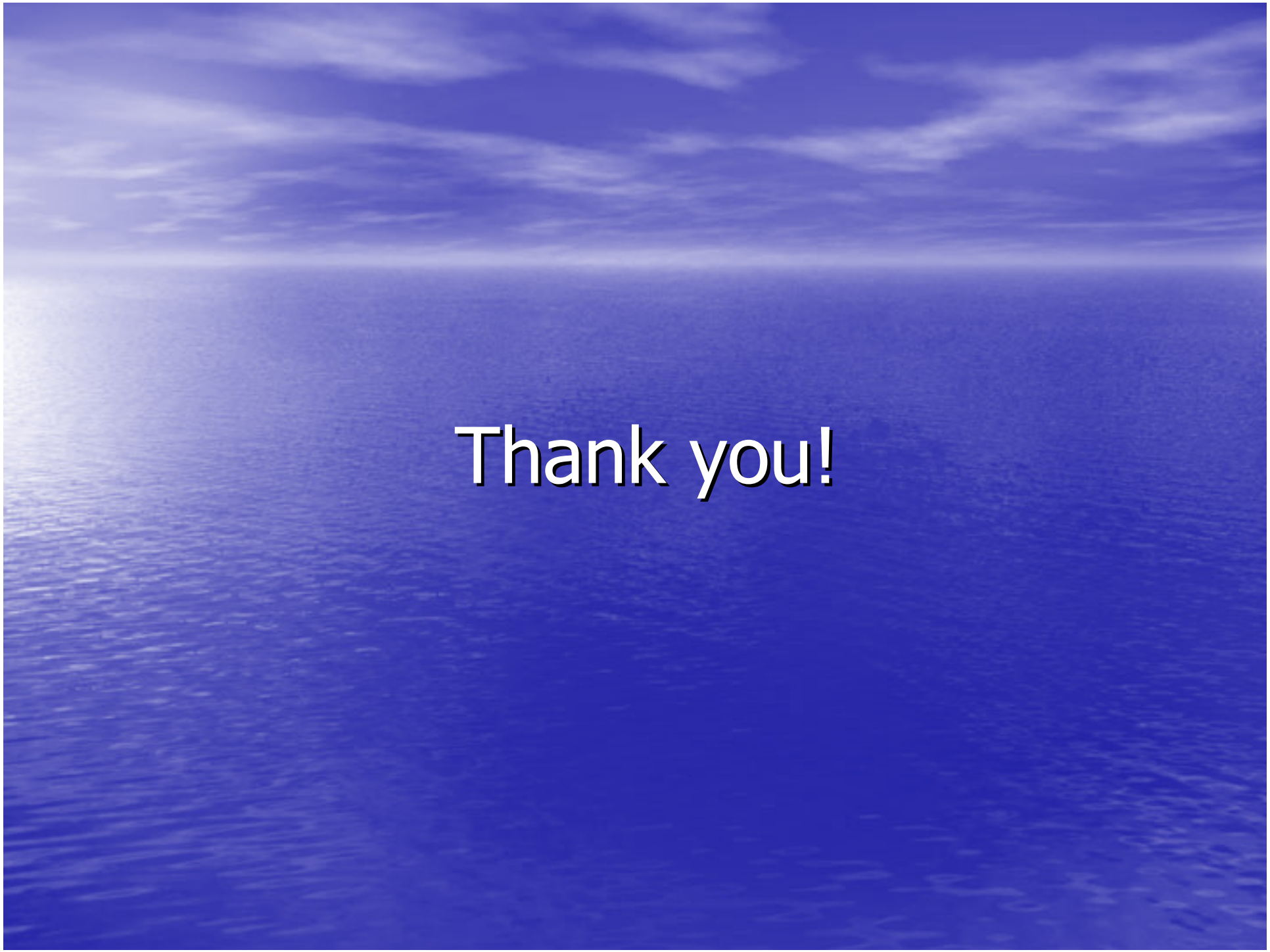


What's next?

- Assess performance on new hardware
- New 'real' TAG Database for CSC data
- Continue gathering data about likely query patterns
- Assess file vs. relational database to guide users as to when each is appropriate
- Trigger implementation in TAGs

Summary

- 1TB tests optimised and measured performance of realistic scale TAG database
- Understand challenges and develop a system that can offer good performance
- Ongoing challenge


Thank you!

Scenarios

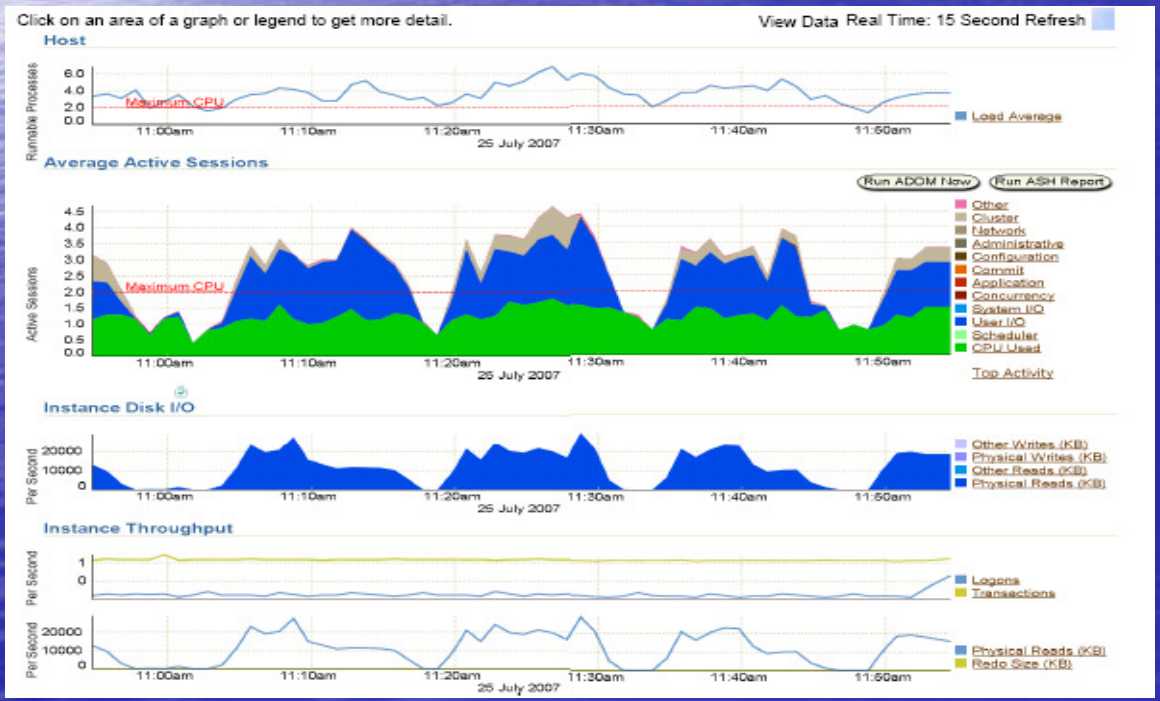
- Complete set of data (or most used part thereof) fits in memory/cache
  - memory is O(1GB)
  - query time is limited by cpu and mem speed
- Complete set of Indices (or most used part thereof) fits in memory/cache
  - fast identification of data
  - slow retrieval from disk
    - contiguous parts @ O(100MB/s)
    - random parts @ O(1000 IO/s) = O(1 MB/s)
    - contiguous IO degrades with parallelism
- Neither Indices nor data fit in memory
  - index reading from disk
    - usually contiguous
- Final and/or intermediate results do not fit in memory
  - e.g. for sorting, intersection, joins, ...
  - need to use disk even for these ops

HASH JOIN, BITMAP CONVERSION TO ROWIDS, BITMAP INDEX RANGE SCAN, INDEX RANGE SCAN, PARALLEL

summary

QUERYINDEX (FULL SCAN) OF 'ICMG1_1_ID', BITMAP INDEX (RANGE SCAN), BITMAP MERGE, BITMAP AND, BITMAP CONVERSION (TO ROWIDS), HASH JOIN, SORT (UNIQUE), QUERY PARALLEL

content



- Each query scans 1Gb of data, which in logical units, would be around 1 hour and 30 minutes of data at 200Hz speed
- Average time of job running alone is 10 minutes
- Node has 2Gb memory and 2 CPUs