IBM Almaden Research Center

It's the PEOPLE, Stupid!

Self-Managing Research at IBM Almaden

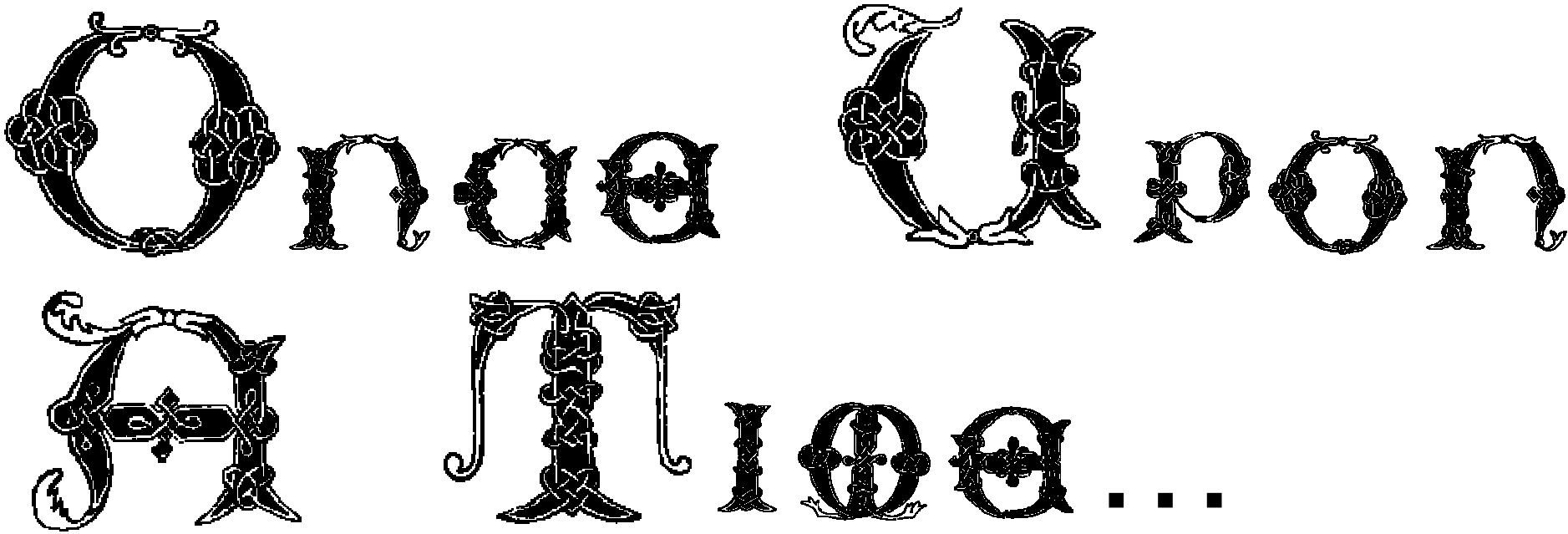
Guy M. Lohman

(lohman@almaden.ibm.com)

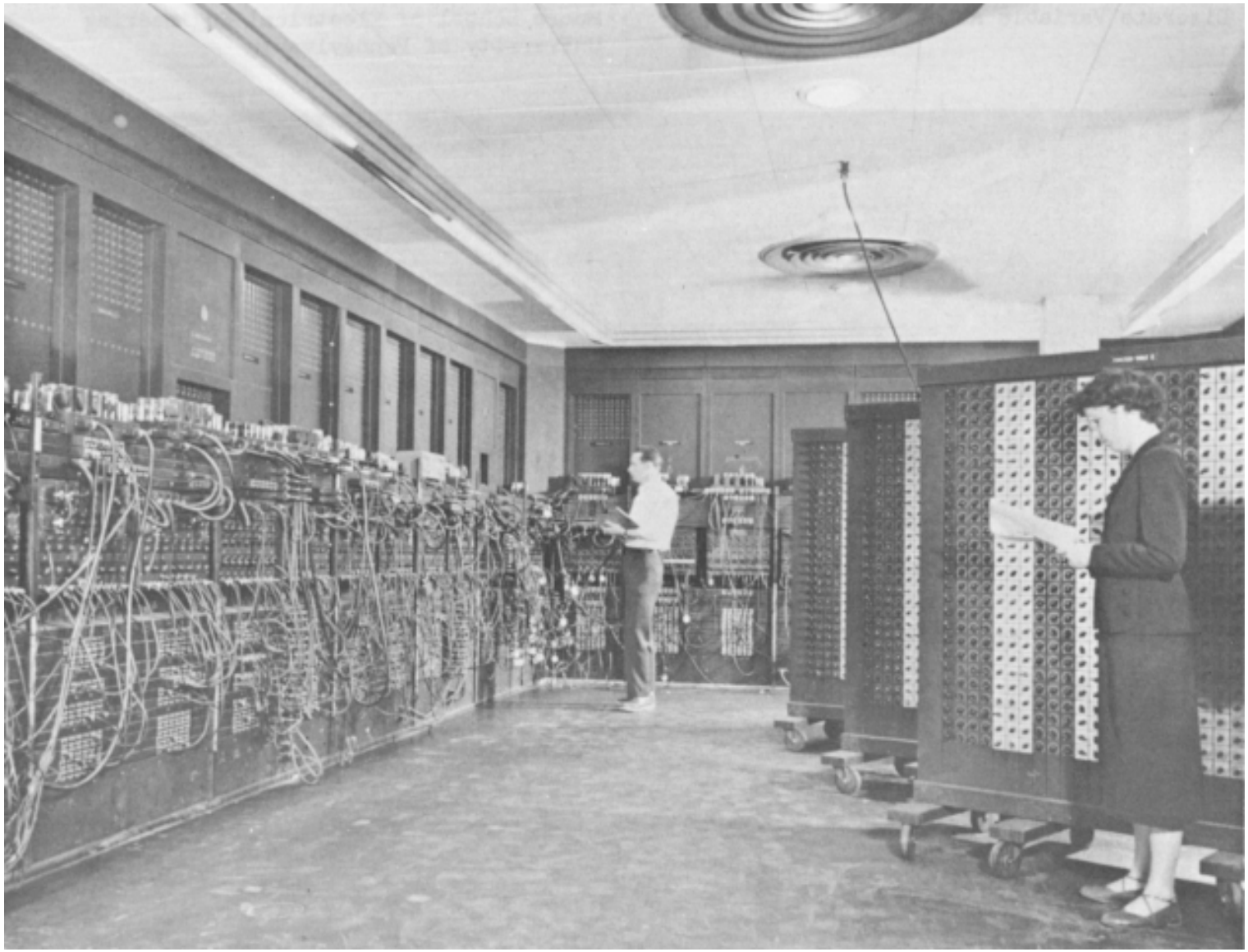
IBM

Agenda

1. **Motivation (history of Comp. Sci.)**
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - Partition Advisor
   - LEarning Optimizer (LEO)
   - Progressive OPtimizer (POP)
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

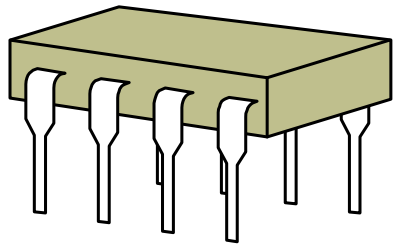


- Computers were
  - Large
  - Unreliable
  - Expensive (Millions of $$$)
- People were cheap (comparatively)

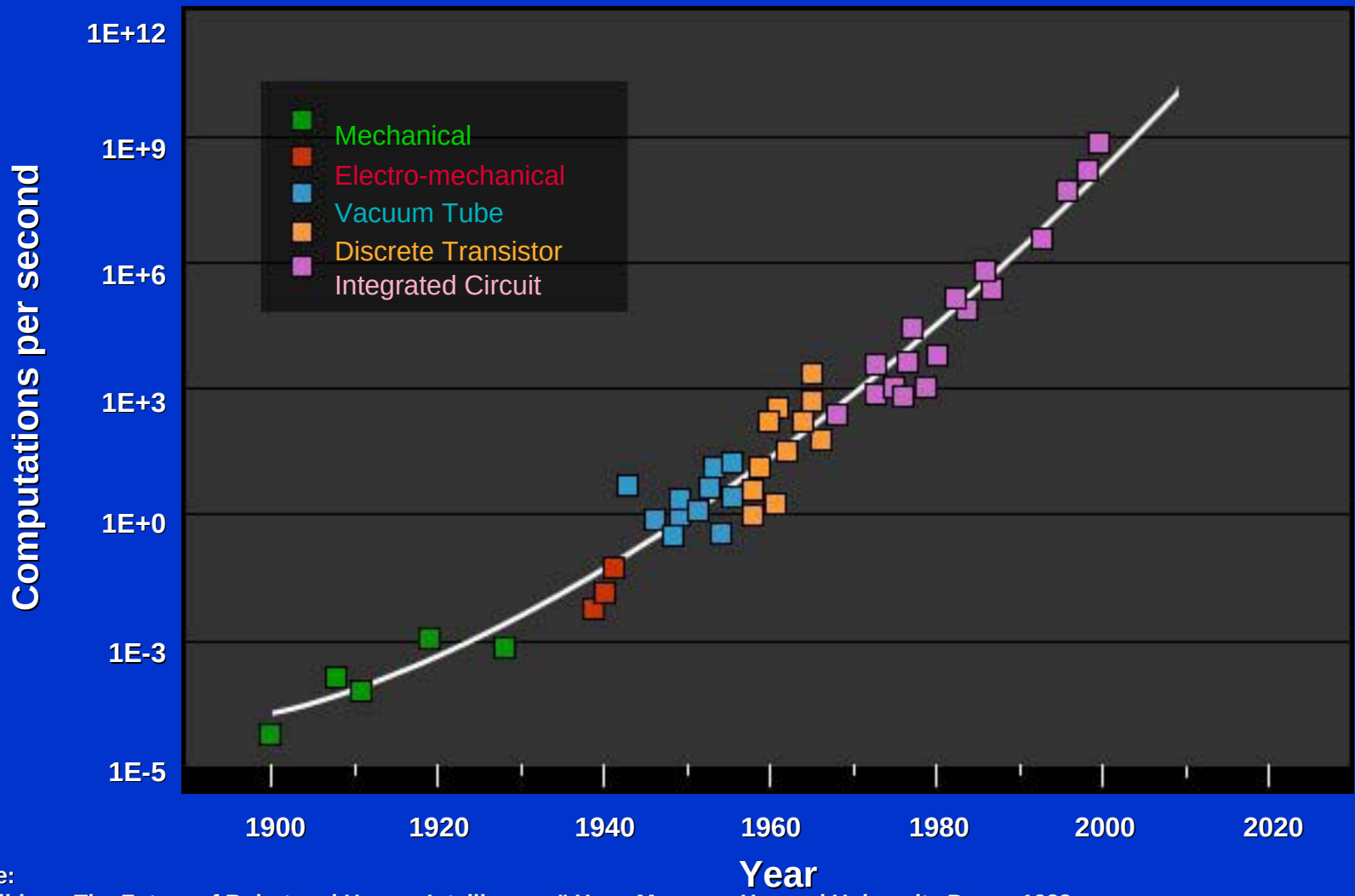


Technology Marches On!

- Chips (CPUs and memory) got
  - Smaller
  - Faster
  - More reliable
  - Cheaper (tens of $)!
- Storage got
  - Bigger
  - Faster
  - More reliable
  - Cheaper (fractions of pennies)!
- Communication got
  - Faster
  - More reliable
  - Cheaper!
- Systems got a lot more complex!
- People got
  - Not appreciably faster or more reliable
  - (a lot) More expensive!

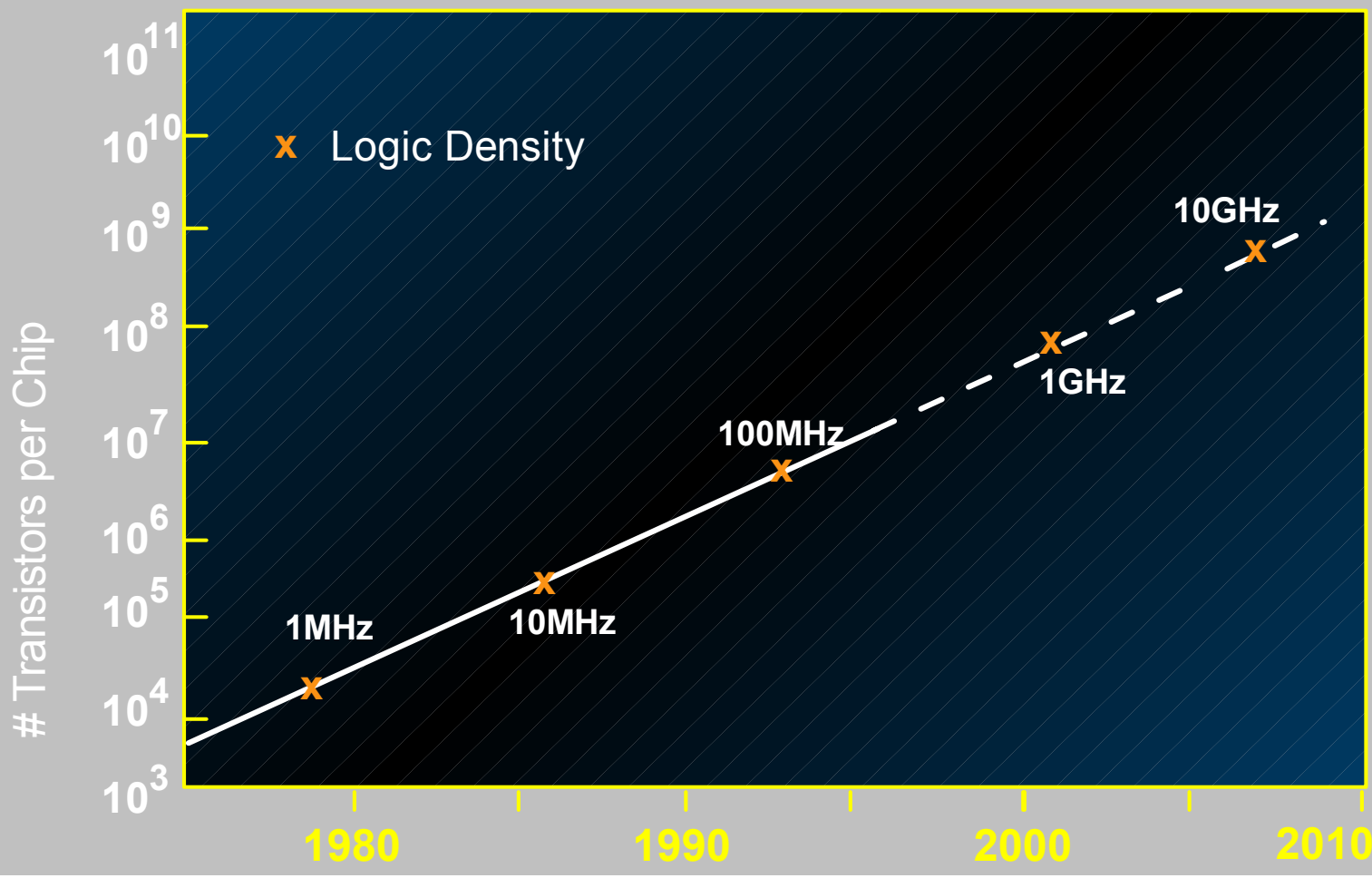


$1000 Buys…



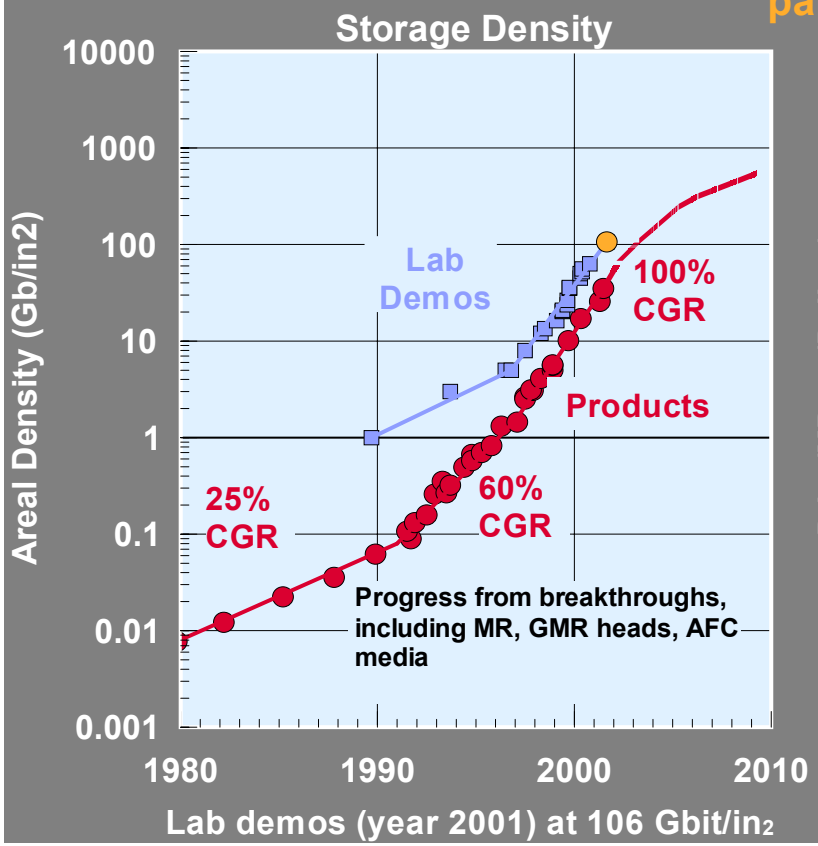
Reference:
"Mind Children: The Future of Robot and Human Intelligence," Hans Moravec, Harvard University Press, 1988,
"The Age of Spritual Machines," Ray Kirzweil, Viking, 1999.

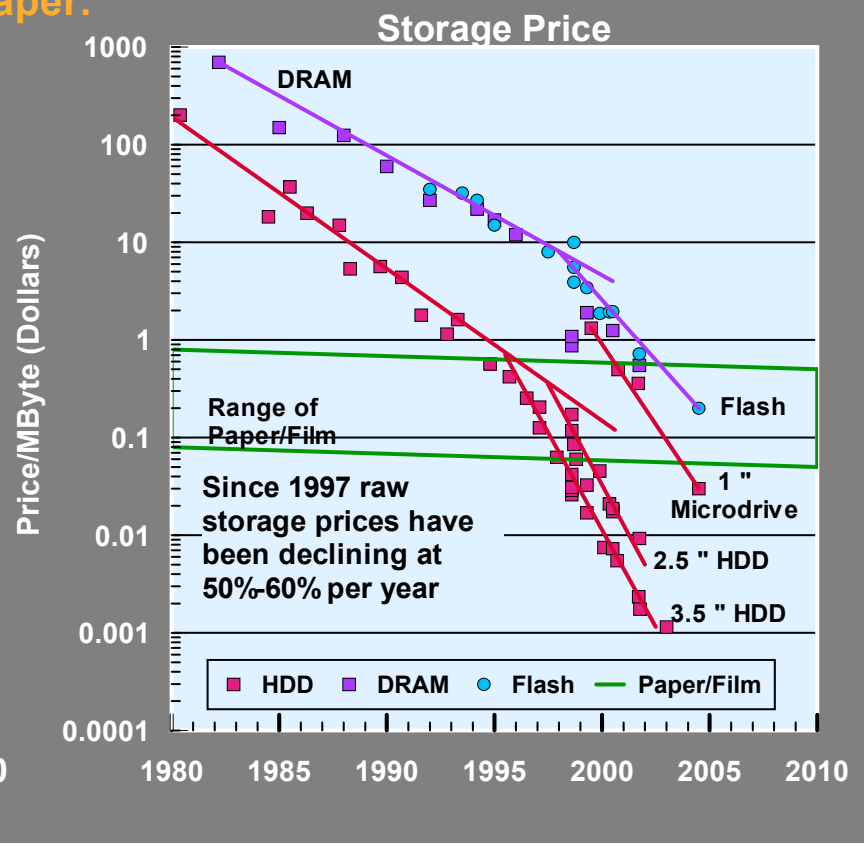
Integrated Circuit Performance Trends



Storage Trends

Storage areal density CGR continues at 100% per year to >100 Gbit/in$_2$. The price of storage is decreasing rapidly, and is now significantly cheaper than paper.





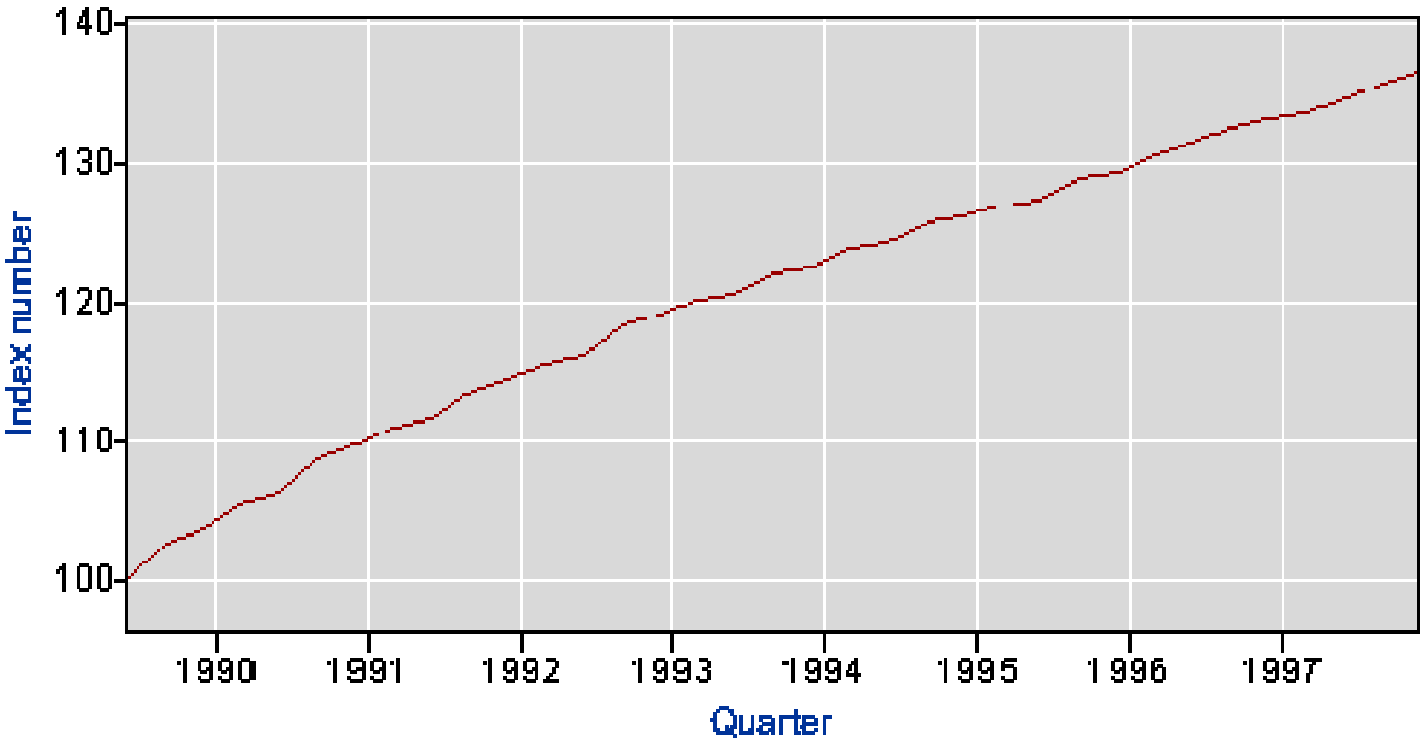
Today

- Disks on laptops have more capacity than most need
  - 1 Terabyte for $1199: http://www.lacie.com/products/product.htm?id=10118
- CPUs cost less than a good meal
  - Complete “bare bones” machines for $200 (retail)
  - Example: http://shop1.outpost.com/product/3847537
- Network capacity glut permits streaming voice and video
- But people, …

Cost of Labor Rarely Decreases (Despite Outsourcing)

Employment Cost Index (1989 = 100):

Total comp., Professional, Specialty, & Technical Occupations (all civilian)

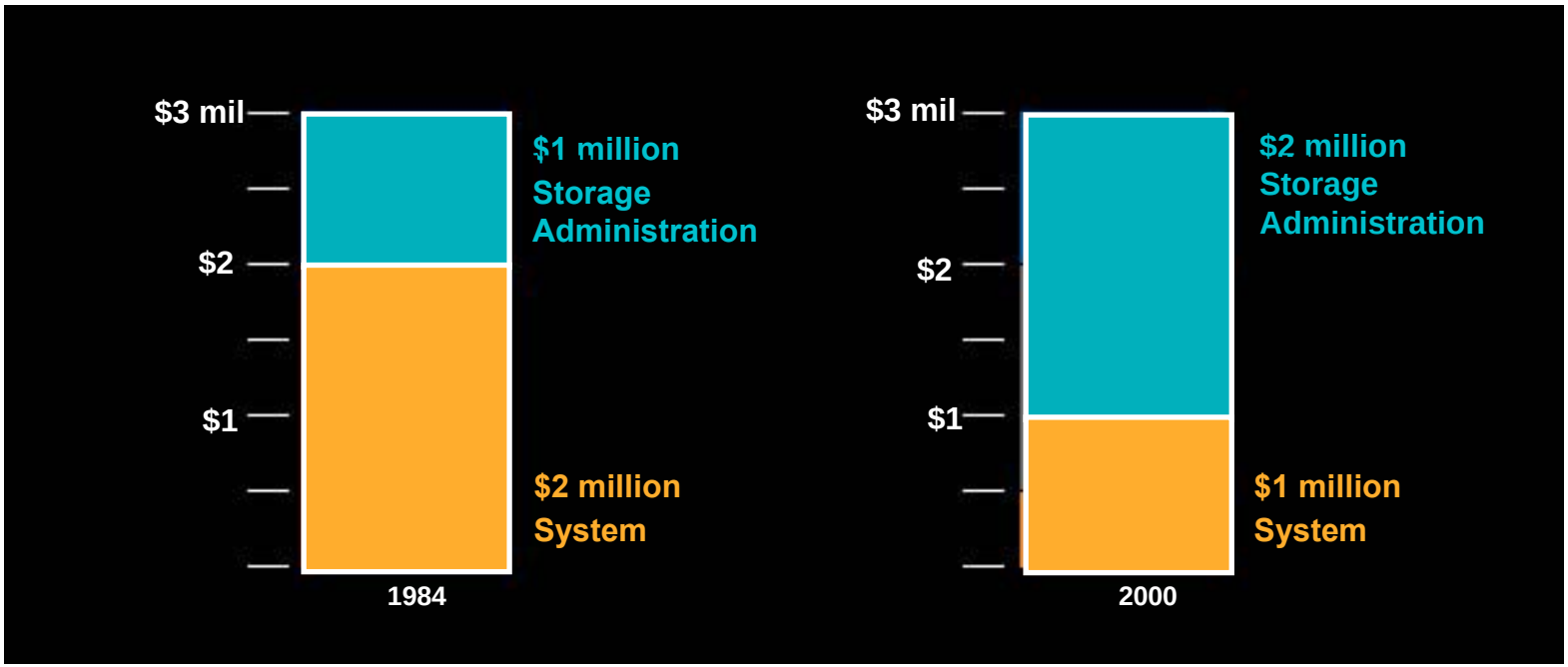


- **Source:** U.S. Bureau of Labor Statistics (http://data.bls.gov/servlet/SurveyOutputServlet)
  Series Id: ECU11121I

THE HIGH COST OF I/T MANAGEMENT

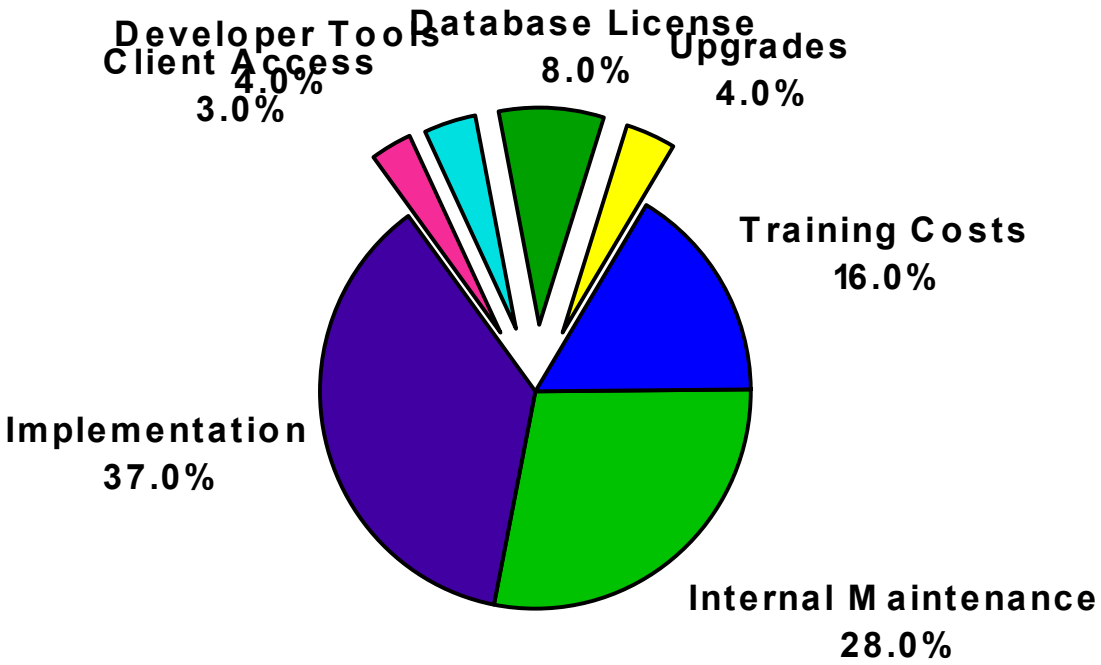
Example: The cost to manage storage is typically twice the cost of the actual storage system!

Storage: What $3 million bought in 1984 and 2000.



(1) J. P. Gelb, "System-managed storage," IBM Systems Journal, Vol 28, No. 1, 1989 pp. 77-103.
(2) "Storage on Tap: Understanding the Business Value of Storage Service Providers", ITCentrix report, March 2001.
(3) "Server Storage and RAID Worldwide" (SRRD-WW-MS-9901), Gartner Group/Dataquest report, May 1999.

Human Costs Dominate in Database, Too

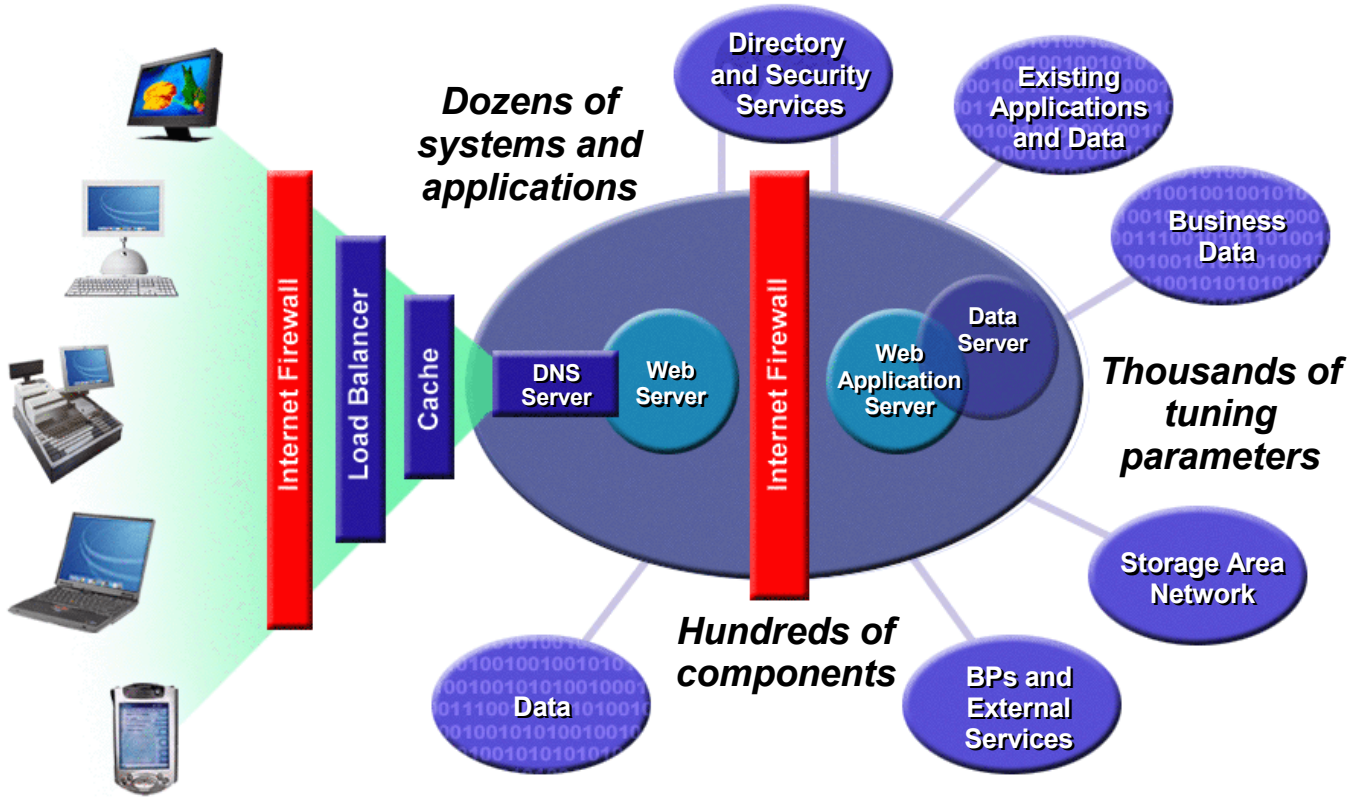


81% is "People Cost"

Source: The AberdeenGroup, 1998
http://relay.bvk.co.yu/progress/aberdeen/aberdeen.htm

Houston, we have a problem …

Complex heterogeneous infrastructures are the norm!

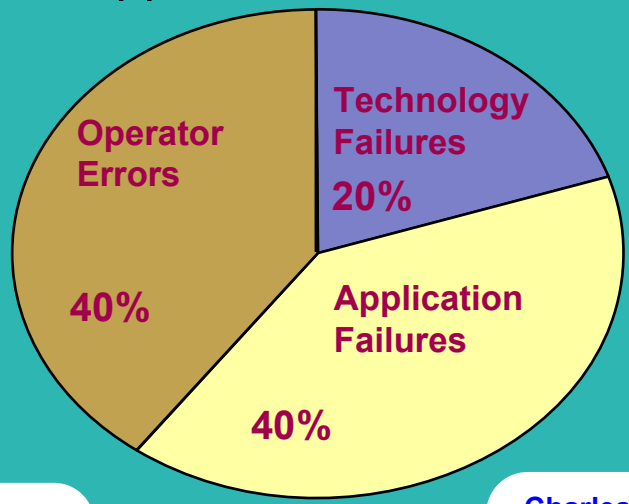


Making the Front Page

eBay
Outage: 22 hours 12 June 1999
Operating System Failure
Cost: $3 million to $5 million revenue hit and 26% decline in stock price

E*Trade
3 February 1999 through 3 March 1999: Four outages of at least five hours
System Upgrades
Cost: ????
22 percent stock price hit on 5 February 1999

Causes of Unplanned Application Downtime



AT&T
13 April 1998 outage: Six to 26 hours
Software Upgrade
Cost: $40 million in rebates
Forced to file SLAs with the FCC (frame relay)

Dev. Bank of Singapore
1 July 1999 to August 1999: Processing Errors
Incorrect debiting of POS due to a system overload
Cost: Embarrassment/loss of integrity; interest charges

America Online
6 August 1996 outage: 24 hours
Maintenance/Human Error
Cost: $3 million in rebates
Investment: ???

Charles Schwab & Co.
24 February 1999 through 21 April 1999: Four outages of at least four hours
Upgrades/Operator Errors
Cost: ???; Announced that it had made $70 million in new infrastructure investment.

Source: Gartner Group

If it is so efficient, why doesn't it fix itself!

"If it's so efficient, why doesn't it fix itself!"

The Goal: A Self-Managing DB2

Wouldn't it be great if your database was as easy to maintain and as self-controlled as your refrigerator?

- **Adjust every configuration parameter dynamically, while the system is in use!**
- **Expand and shrink memory usage, based on workload**
- **Automatically profile workloads and create/recommend indexes, partitioning, clustering, summary tables, ... to improve performance**
- **Automatically detect the need, estimate the duration of, and schedule maintenance operations**

(like REORG, statistics collection, backup, load, rebind)

- **Observe actual performance and exploit that information to improve operations.**
- **Recommend action when things aren't the way you want them to be.**
- **Project into the future to detect coming problems, like low memory or constrained disk space, and notify you by page or e-mail, days or weeks in advance!**

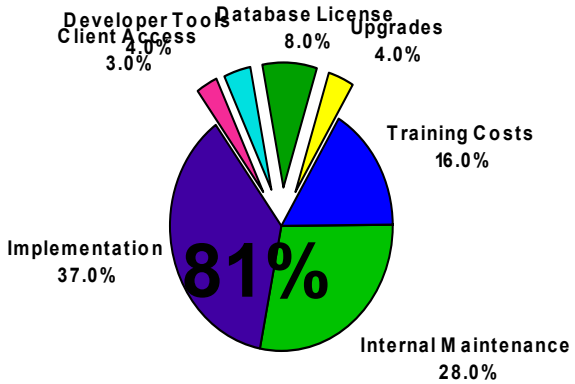
Motivation

- Complexity: it isn't getting any easier!!
  - More & more DBs, tables, users: 100s ---> 10,000s
  - Large applications, GBs & TBs of data, clusters of servers
  - Need to keep 1000s of users connected to 100s of DBs
- Who's gonna set it up and keep it running?
  - Skilled DBAs are increasingly rare
  - ISVs, .COMs want embedded, invisible DBs
  - Smaller shops don't have specialized skills, must diversify

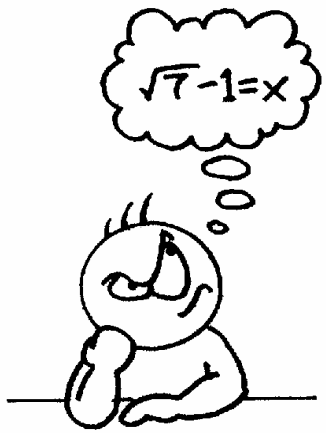
Motivation

- Complexity: it isn't getting any easier!!
  - More & more DBs, tables, users: 100s ---> 10,000s
  - Large applications, GBs & TBs of data, clusters of servers
  - Need to keep 1000s of users connected to 100s of DBs
- Who's gonna set it up and keep it running?
  - Skilled DBAs are increasingly rare
  - ISVs, .COMs want embedded, invisible DBs
  - Smaller shops don't have specialized skills, must diversify

Server TCO:



Source: The AberdeenGroup, 1998
http://relay.bvk.co.yu/progress/aberdeen/aberdeen.htm



Agenda

1. Motivation
2. **Self-Optimizing (DB2)**
   - **Design Advisor**
     - Index Advisor
     - Partition Advisor
   - LEarning Optimizer (LEO)
   - Progressive OPtimizer (POP)
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

The Design Advisor

- An extension of existing Index Advisor (V6, 1999)

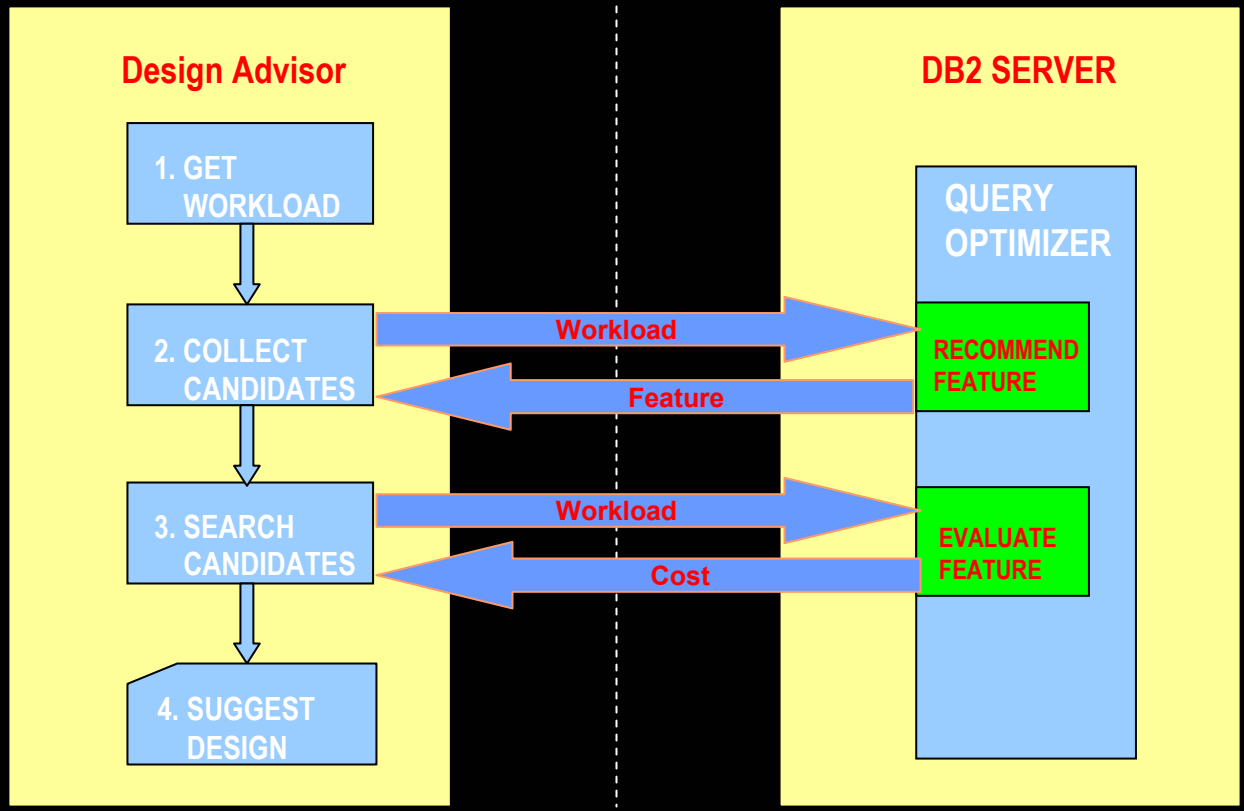
- Headquarters for all physical database design, including:
  - ✓ Partitioning of tables in partitioned environment
  - ✓ Materialized views
    - ➢ Now extended to Materialized Query Tables (MQTs)
    - ➢ Called Automatic Summary Tables (ASTs) before V8
  - ✓ Multi-Dimensional Clustering (MDC) storage method
  - ✓ Others in future?

- Recommends any combination of the above aspects

- Status: Shipped in DB2 Universal Data Base V8.2 (Sept. 2004)

- **Refn.: "DB2 Design Advisor: Integrated Automatic Physical Database Design", VLDB 2004 (Toronto), Zilio, Rao, Lightstone, Lohman, et al.**

Autonomic Database Design: the DB2 Design Advisor



- First-ever integrated "One Stop Shopping" for Physical Database Design
  - Indexes
  - Materialized Query Tables
  - Partitioning
  - Clustering
- Fully integrated with DB2 Optimizer as "What If?" Engine

Index Advisor (DB2 V6) – The Math

- Variant of well-known "Knapsack" Problem
- Greedy "bang-for-buck" solution is optimal,
  when integrality of objects (indexes) is relaxed

- **For each query Q**:
  - Baseline: Explain each query w/ existing indexes, to get cost E(Q)
  - Unconstrained: Explain each query in **RECOMMEND INDEXES** mode, to get cost U(Q)
  - Improvement ("benefit") B(Q) = E(Q) - U(Q)
- **For each index I** used by one or more queries:
  - If query Q used index I, assign "benefit" B(Q) to index I:
    B(I) = B(I) + B(Q)
  - Assign "cost" C(I) = size of index in bytes
  - Order indexes by decreasing B(I) / C(I) ("bang for buck")
  - Cut off where cumulative C(I) exceeds disk budget
- **(Iterative improvement) While time limit not expired**:
  - Exchange handfuls of "winners" with "losers"
  - Explain each query in **EVALUATE INDEXES** mode

● **Refn.: "DB2 Advisor: An Optimizer Smart Enough to Recommend its Own Indexes", ICDE 2000 (San Diego), Valentin, Zuliani, Zilio, Lohman, et al.**

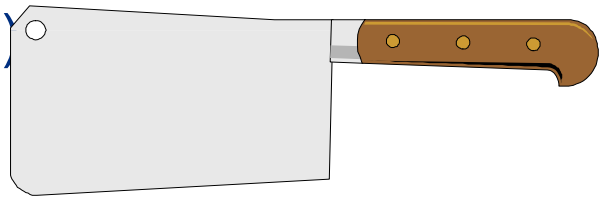
Agenda

1. Motivation
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - **Partition Advisor**
   - LEarning Optimizer (LEO)
   - Progressive OPtimizer (POP)
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

Design Advisor: Partitioning Advisor

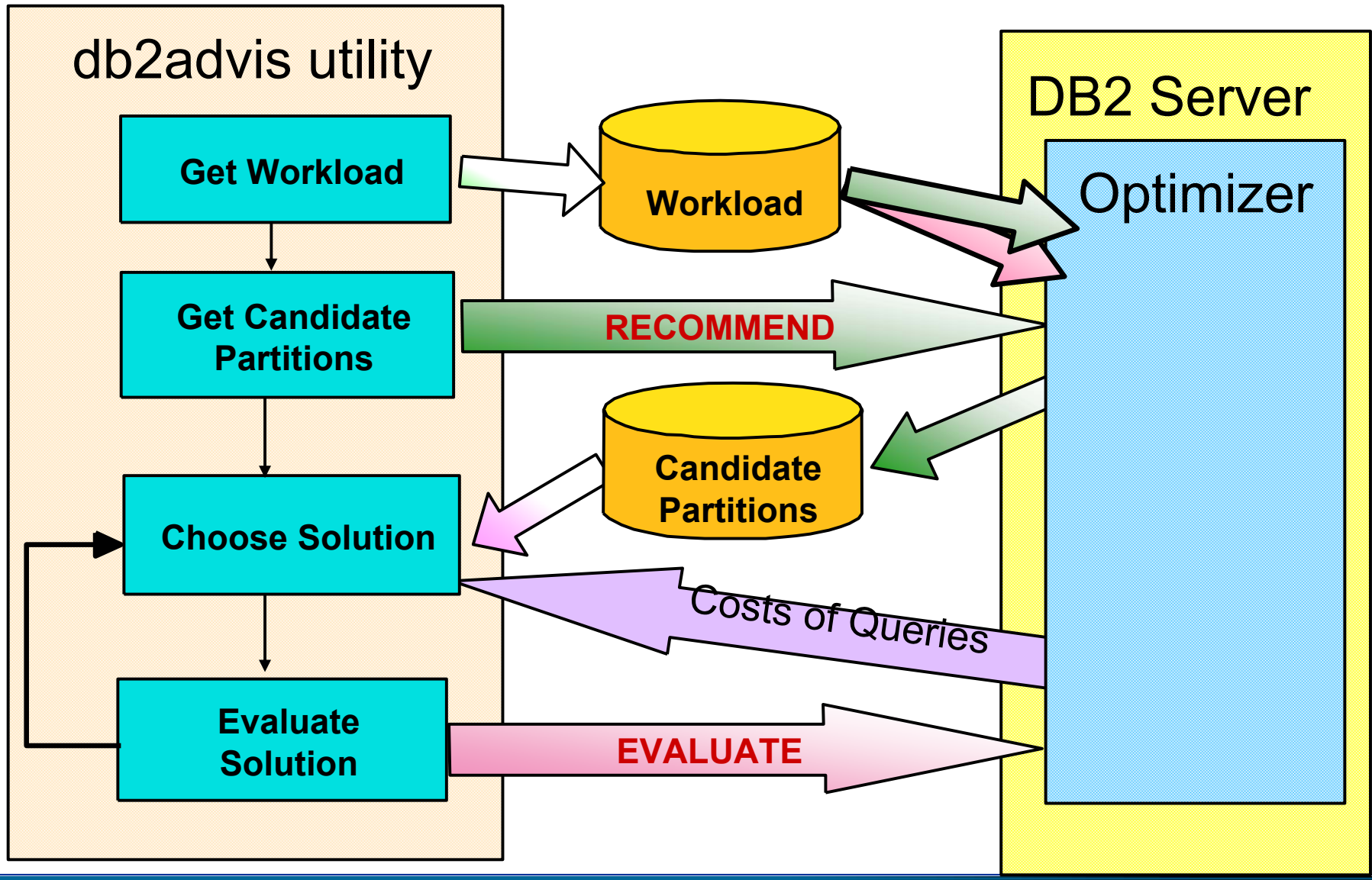
(Jun Rao, Chun Zhang)

- **Scope:**
  - DB2 with Data Partitioning Feature (DPF)
  - "Shared-nothing" parallelism
  - Data stored horizontally partitioned
    - **In a partition group, spread across specified partitions**
    - **Based upon hashing of *partitioning column(s)***
    - **May be replicated across all partitions of partition group**
  - Need to co-locate similar values for joins, aggregation in queries
  - Partitioning required for a given table may be different
    - **Between queries**
    - **Even within a query (joined on different columns)!**
- **Problem:** What is optimal partitioning for each table, given:
  - **Workload of queries**
  - **Schema, including set of partition groups & tablespaces**
  - **Statistics on database**



- **Refn.: "Automating Physical Database Design in a Parallel Database", SIGMOD 2002 (Madison, WI), Rao, Zhang, Lohman, Megiddo.**

Partitioning Advisor Design



RECOMMEND PARTITION Mode

- **Goal: Find best partitioning**
  - For each Table…
  - In each Query in the Workload
- **Approach: Optimizer...**
  - Analyzes query to suggest good "virtual" partitions
  - Generates all combinations of candidate partitions
  - Does its normal plan selection to choose best plan
  - Writes to ADVISE_PARTITION table the partitions of the winning plan

Candidate Partitions Considered

- **Partition groups:**
  - Default partition group (all partitions but catalog partition)
  - Any existing partition groups
- **Partition Columns:**
  - All "interesting" partitions, beneficial to this query for:
    - Joins
    - Aggregation
  - Partitions exploitable by simple, equality local predicates, e.g.

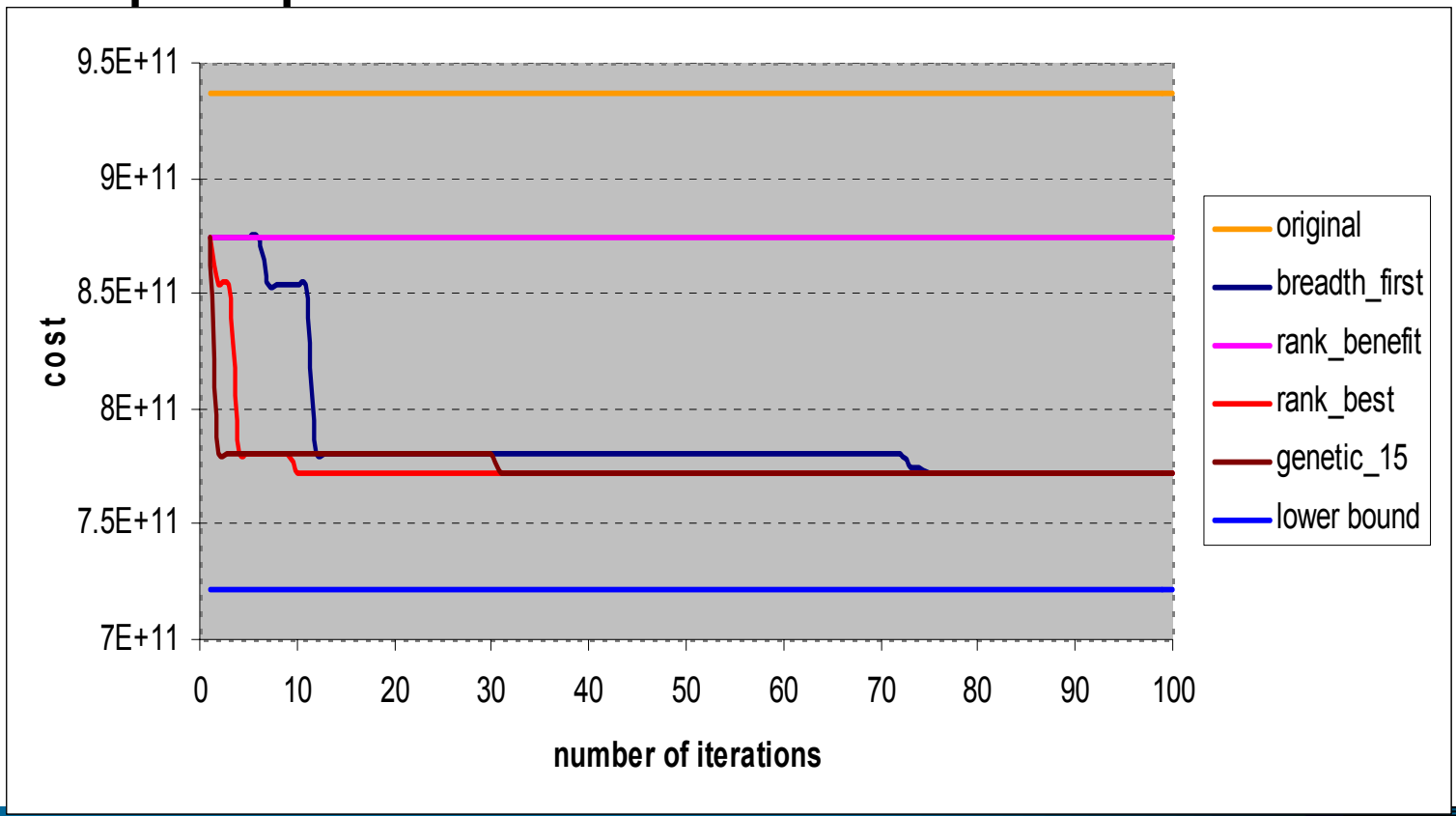
    WHERE keyid = 'ABC'
  - Constraint: Must be subset of any key (unique) columns
- **Replication partitions, for relatively small tables**

EVALUATE PARTITION Mode

- "What if?" mode, to try one solution for all queries in the workload
- Optimizer, for each query Q:
  - For each table T in Q,...
    - reads current partition solution for T, marked in table ADVISE_PARTITION
    - replaces real partition for T with "virtual" partition read.
  - Does its normal plan selection for Q, to get cost of workload
- If cumulative cost for all queries is cheaper than previous solution, db2advis retains that solution

Performance Improvement over Time

- Customer database
- 50 queries and 500 possible configurations
- Rank_best algorithm converges the fastest,
- 22% speedup



Agenda

1. Motivation
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - Partition Advisor
   - **LEarning Optimizer (LEO)**
   - Progressive OPtimizer (POP)
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

LEO: DB2's LEarning Optimizer

(Volker Markl, Ashraf Aboulnaga, Vijayshankar Raman, Alberto Lerner)

I can't believe I did that...!

Reference: ***"LEO: DB2's LEarning Optimizer", VLDB 2001 (Rome), Stillger, Lohman, Markl, Khandil.***

LEO: Overview

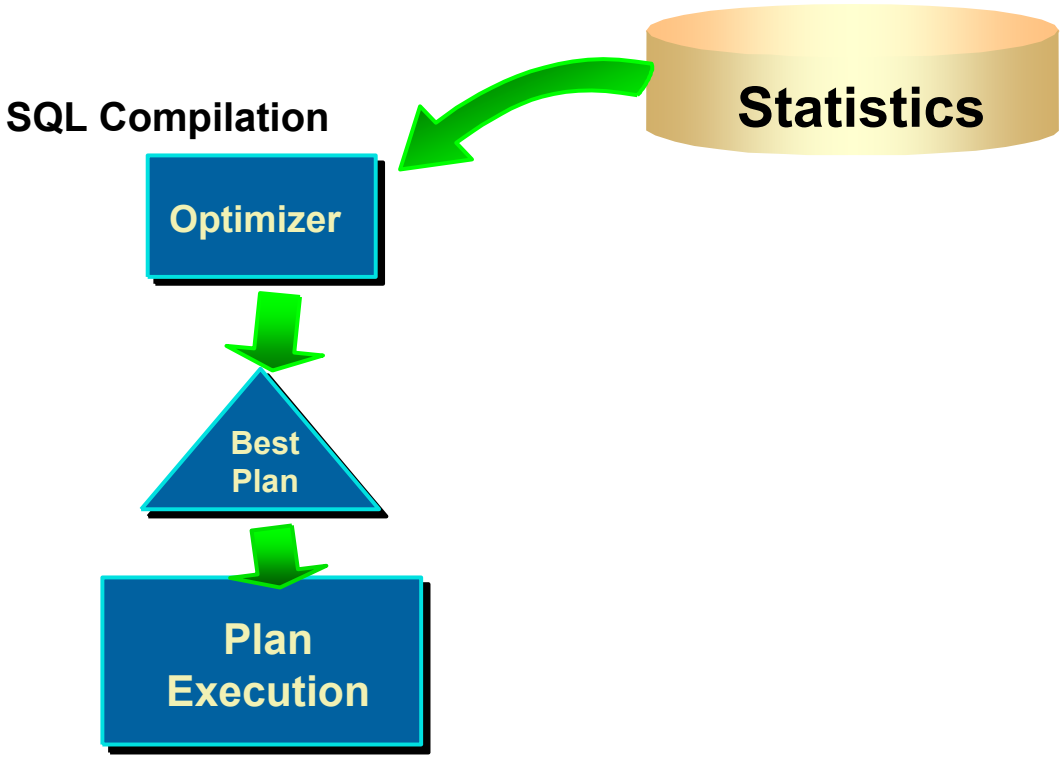
- Cost estimates depend heavily on cardinality estimates
- Cardinality estimates can occasionally be flaky
  - Especially after joins
  - Due to
    - ✓ Correlation between columns
    - ✓ Other things not modeled
    - ✓ Modeling assumptions that may be false
- Why not use empirical results from executed queries to
  - Validate statistics and assumptions
  - Advise when/how to run RUNSTATS
  - Gather statistics that reflect the workload
  - Repair costing in next query optimization round
- Could achieve automatically
  - + Better quality plans
  - + Reduced customer admin. time
  - + Reduced IBM support time
- Status:
  - + Shipped in DB2 UDB V8.2 (September 2004)
  - + **Part of fully automated RUNSTATS**

The Big Win -- Correlation

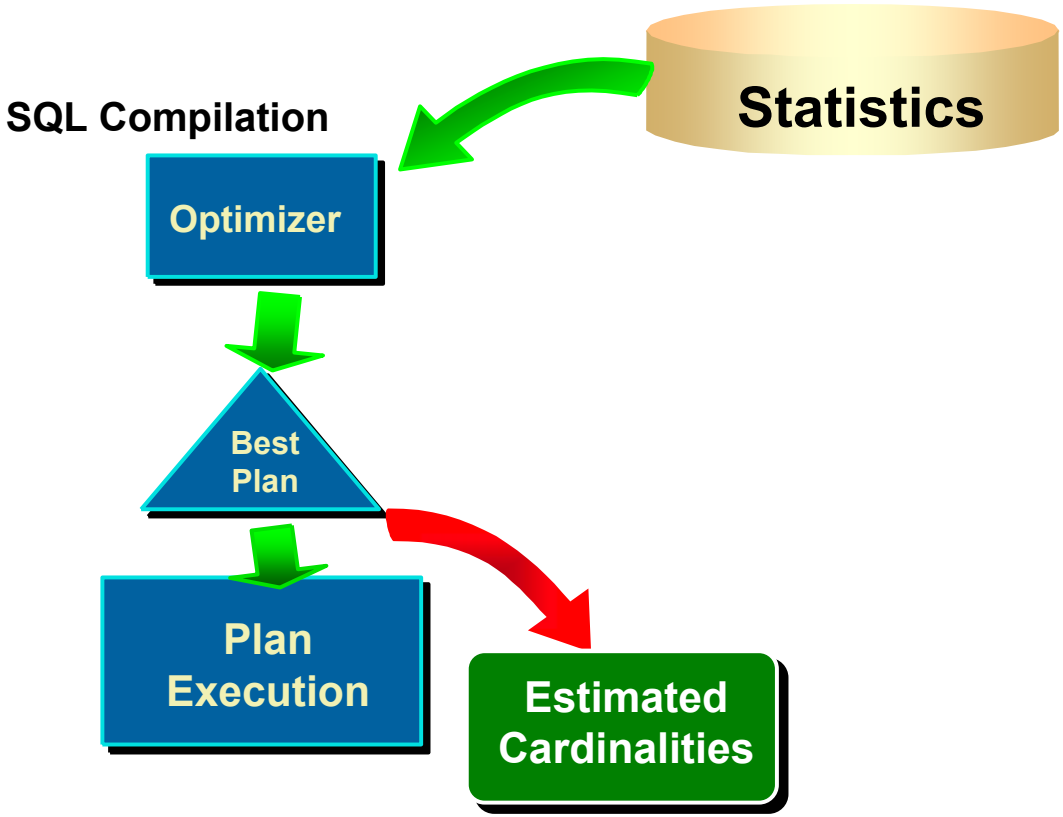
- **EXAMPLE:**
  - WHERE make = 'Honda' AND model = 'Accord'
  - Suppose
    - 10 makes ==> selectivity(make) = 1/10
    - 100 models ==> selectivity(model) = 1/100
  - So selectivity of both = 1/10 * 1/100 = 1/1000
  - But only Honda makes an Accord model!
  - We assumed the predicates were independent by multiplying!
  - In fact, model functionally determines make (predicate on make really adds no information)!
  - Effect: We underestimate cardinality by an order of magnitude!

- **In general,**
  - Can be any subset of predicates in the WHERE clause
  - How do we know which subset of predicates caused the error?
  - How do we generalize to all instances of make and model?

Today

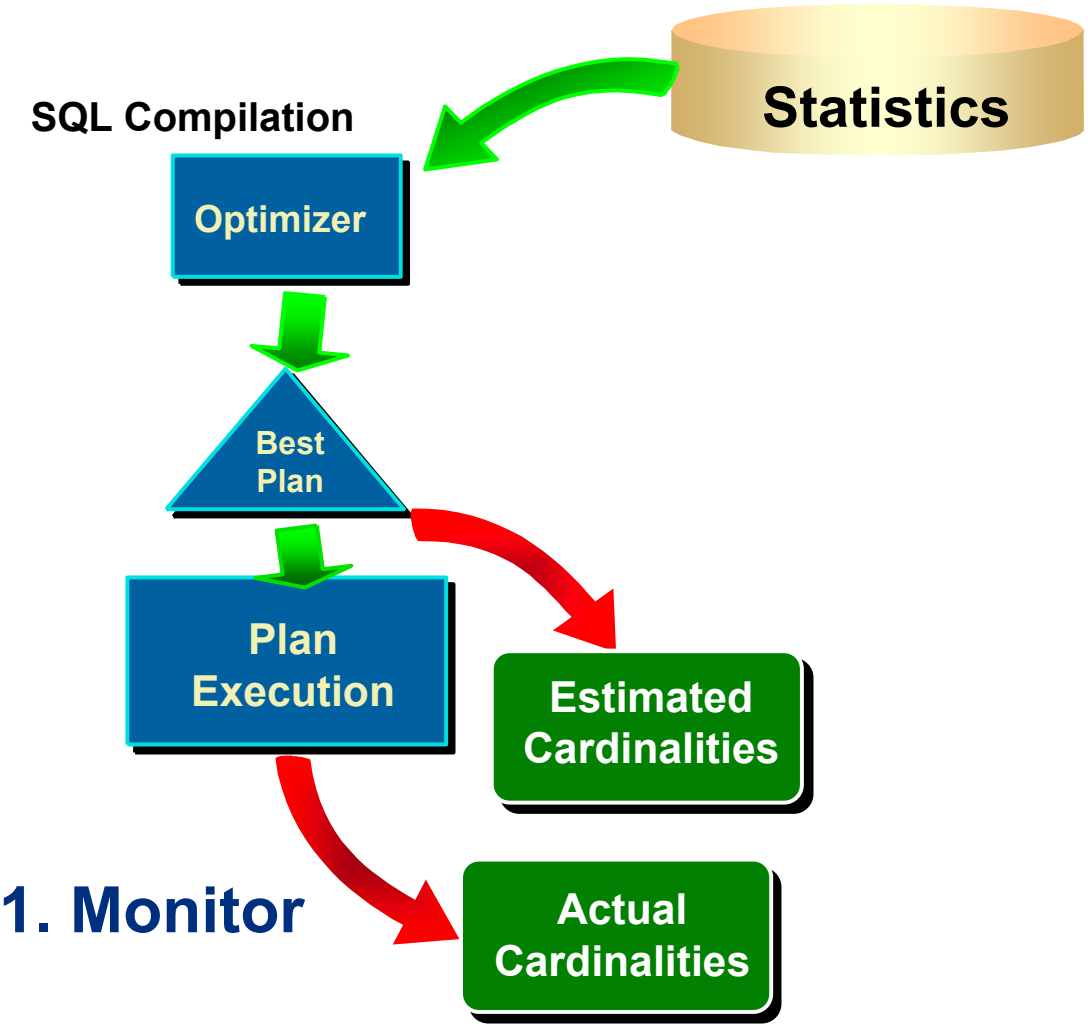


EXPLAIN Gives Estimates Only

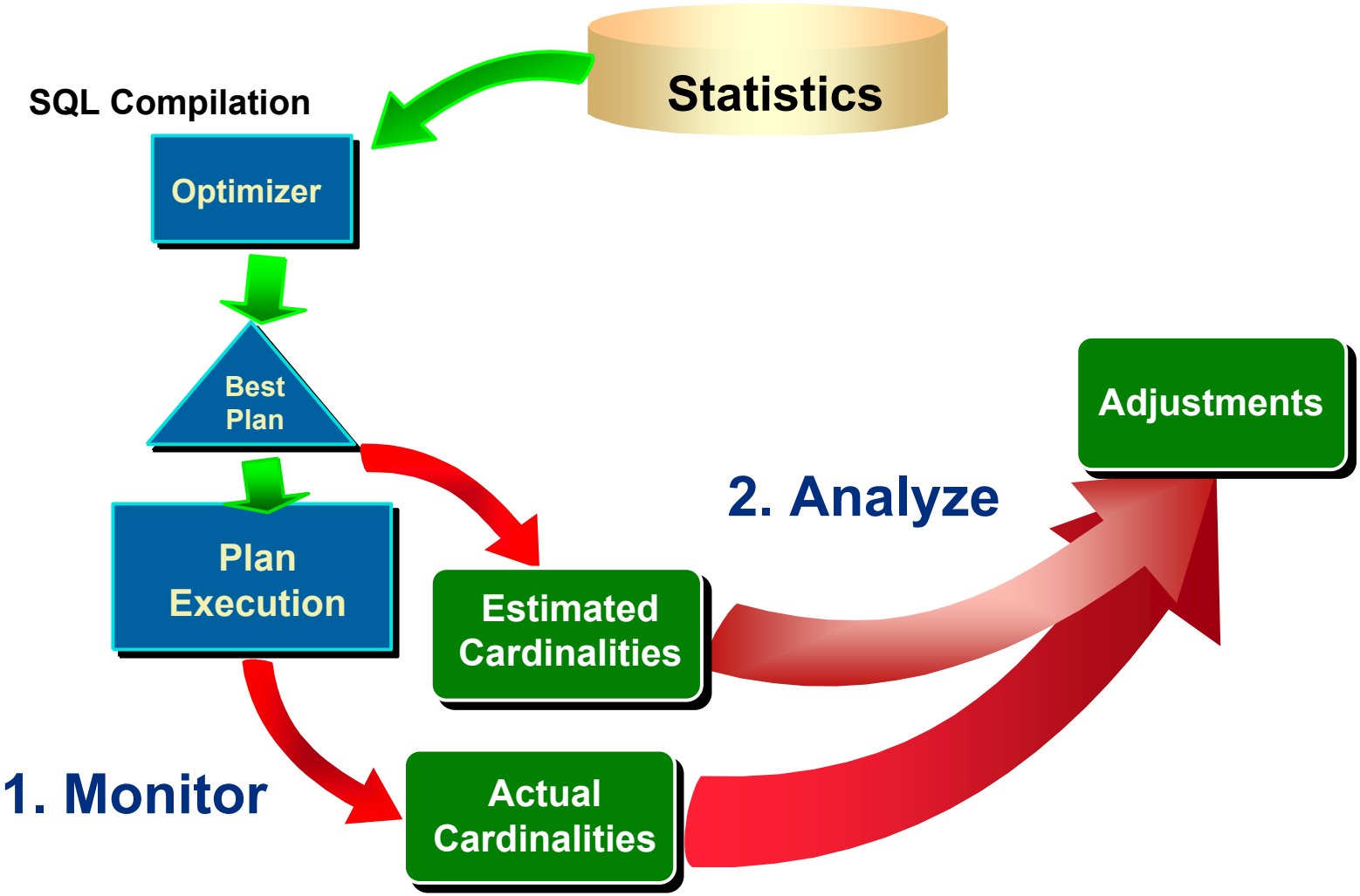


1. Monitor

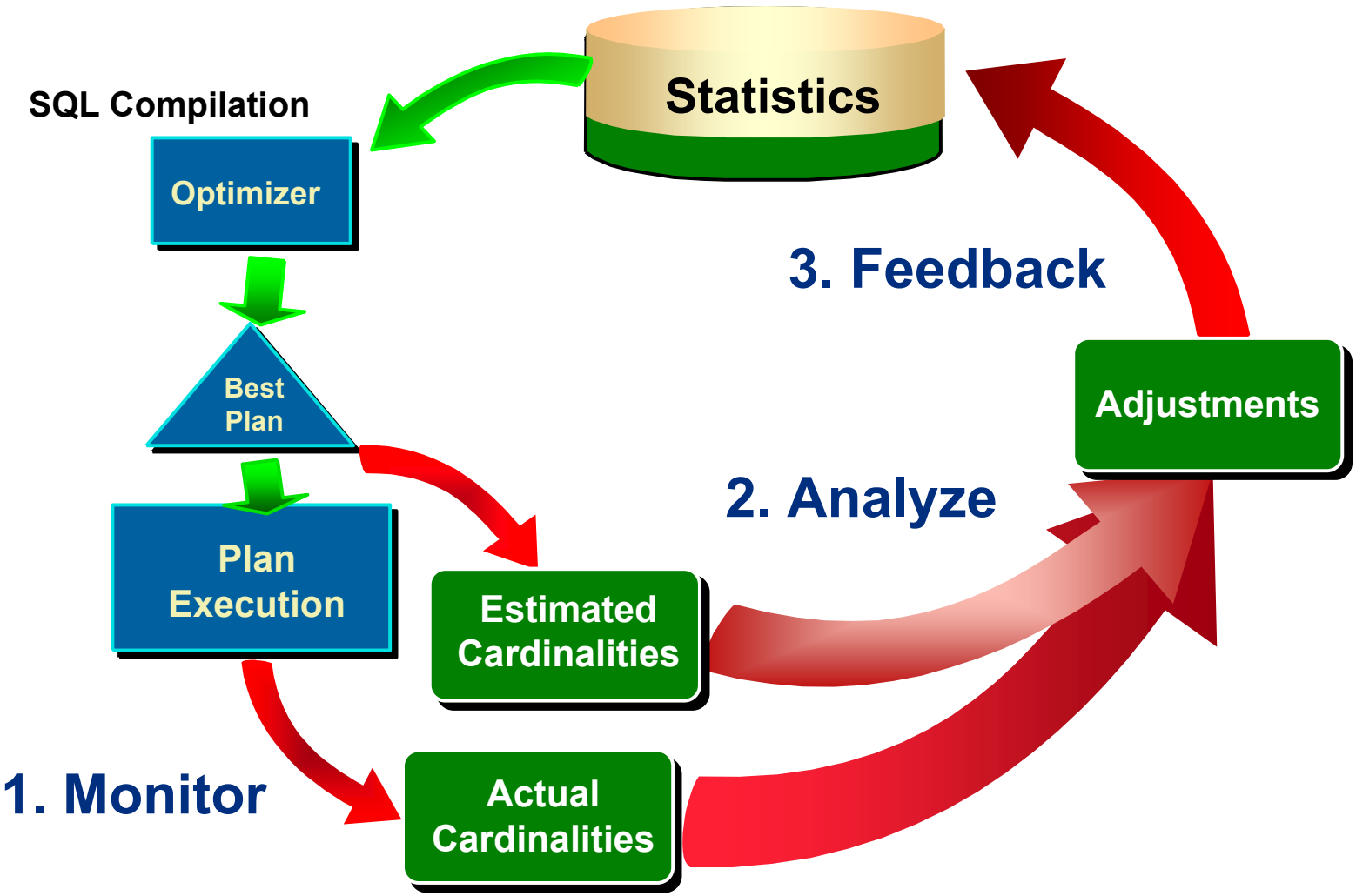
LEO Captures Actual Counts



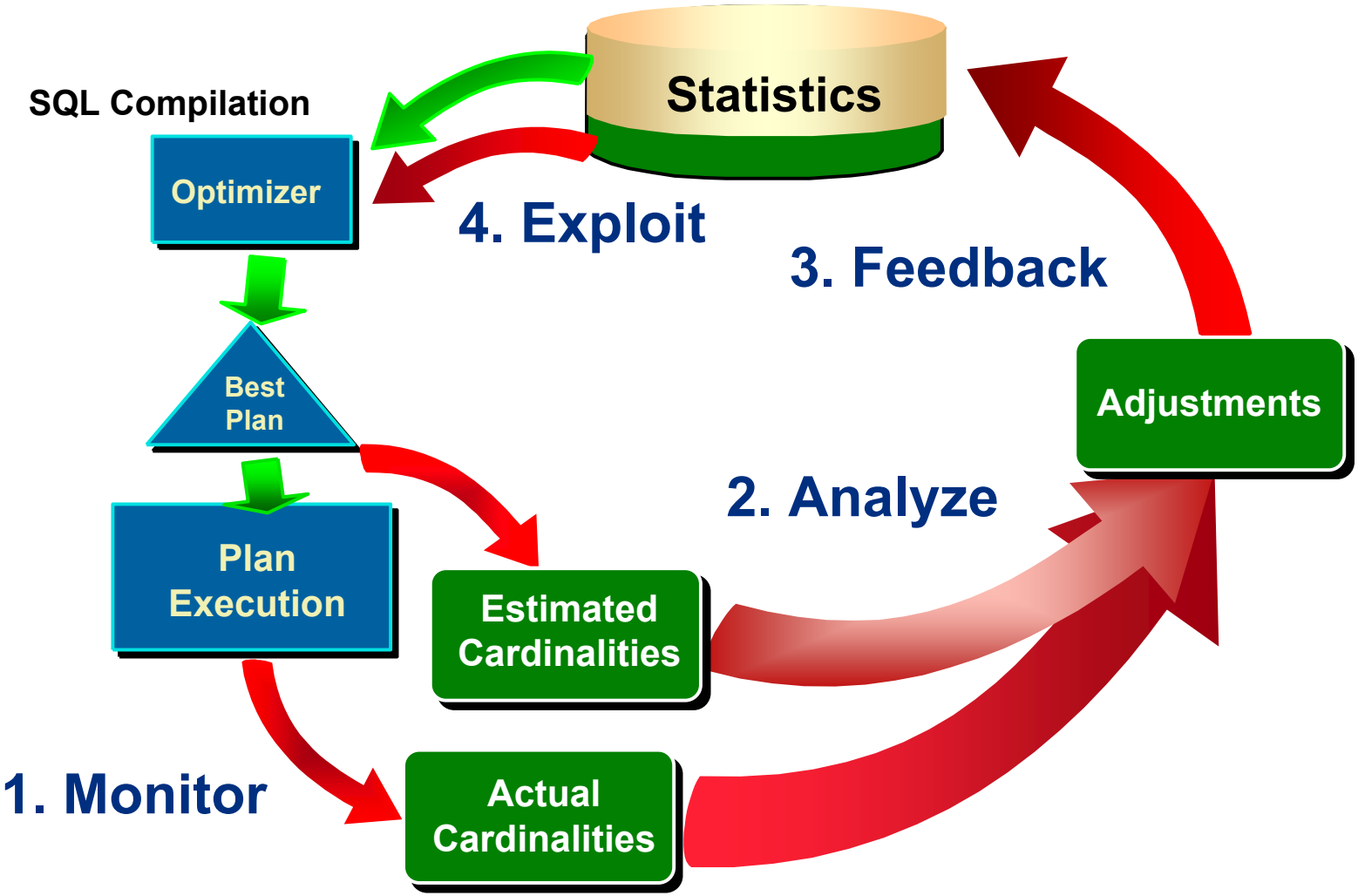
Where Did I Go Wrong?



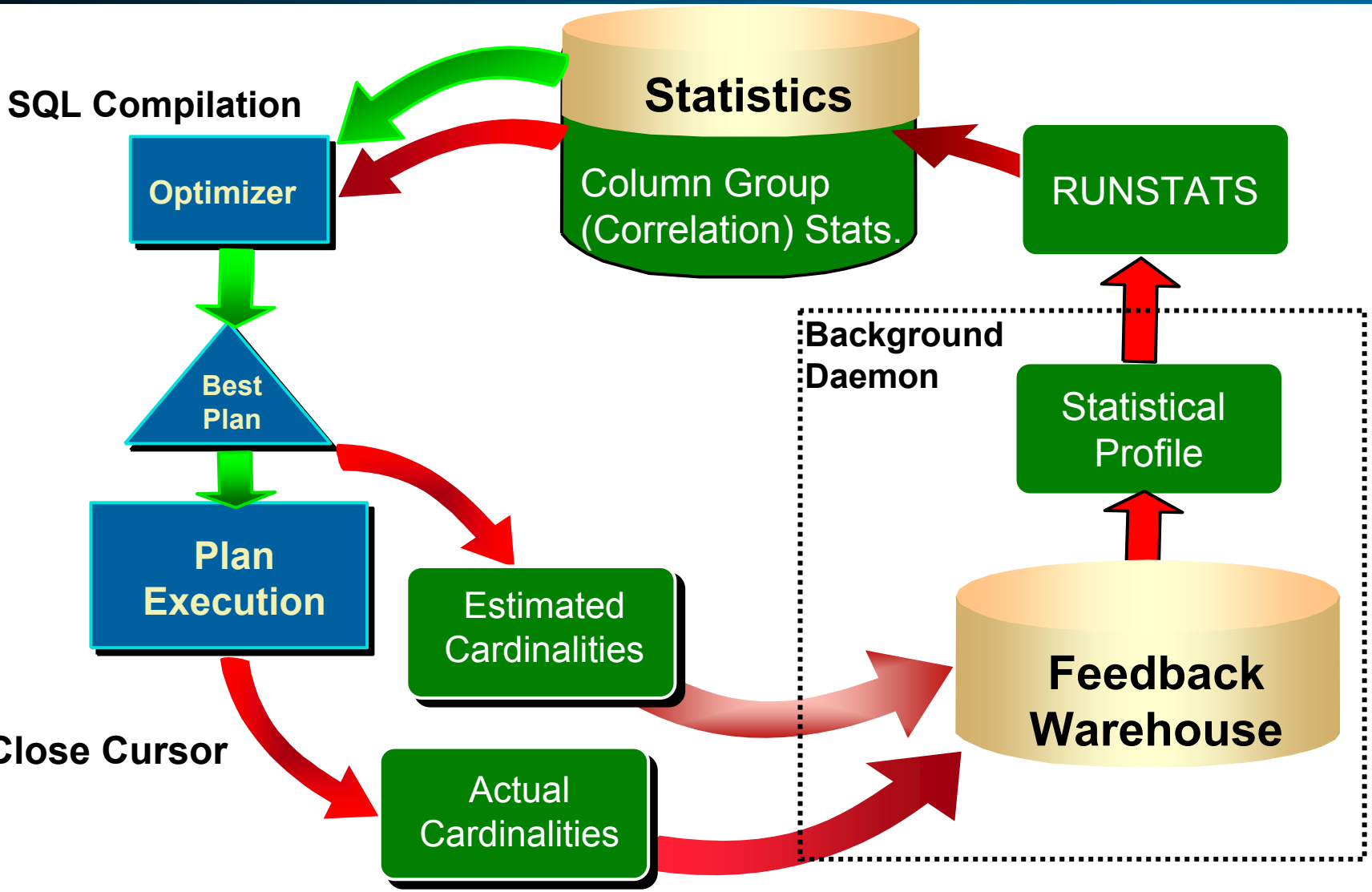
Feedback to Statistics



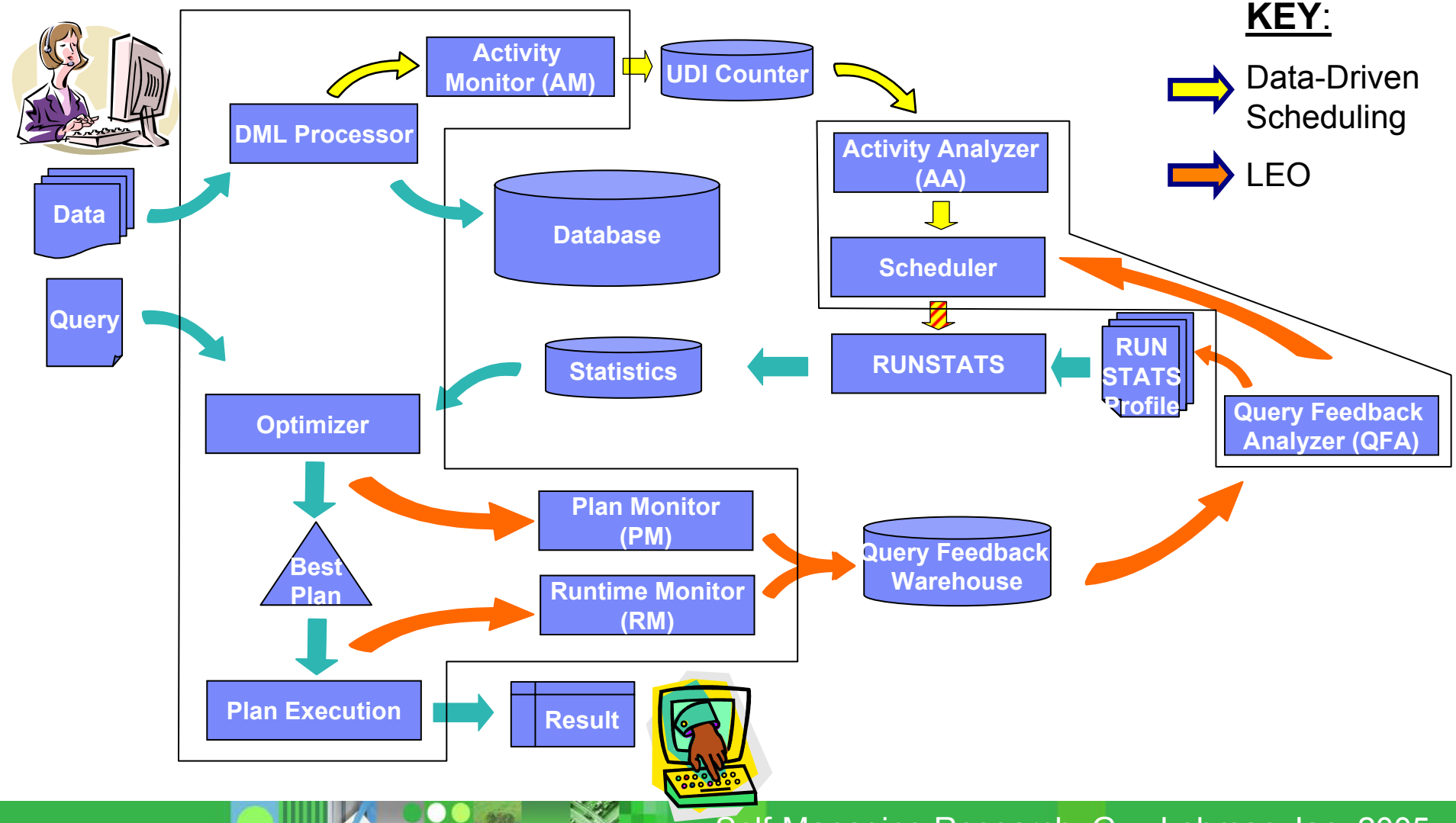
Learning in Query Optimization!



V8.2 (LEO Phase 1) – Directs RUNSTATS

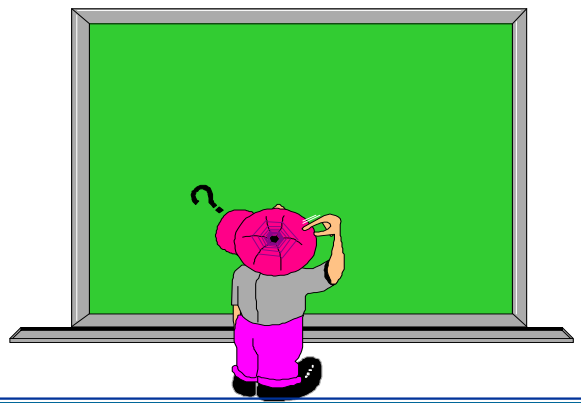


Overview of Automated Statistics Collection



LEO: Some Other Research Issues

- How good/complete is the information we have?
- How do we ensure consistency among data collected at various times?
- How often should we rebind plans?
  - Better information from LEO should improve plans, BUT
    - Don't want to incur compilation cost often
    - Could get a worse plan
    - May want to "lock down" certain plans
- How do we ensure stability (convergence) eventually?
- When will we know that we've collected enough information?

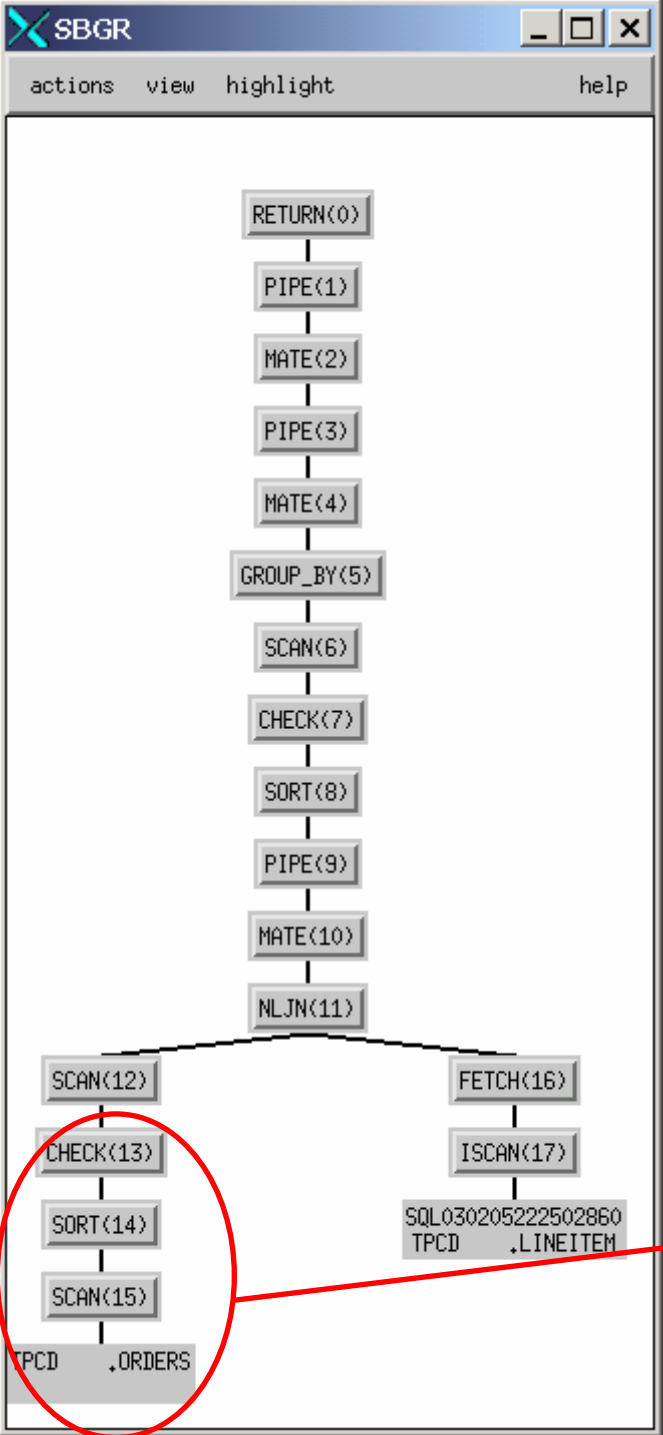


Agenda

1. Motivation
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - Partition Advisor
   - LEarning Optimizer (LEO)
   - **Progressive OPtimizer (POP)**
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

Progressive OPtimization (POP): Motivation

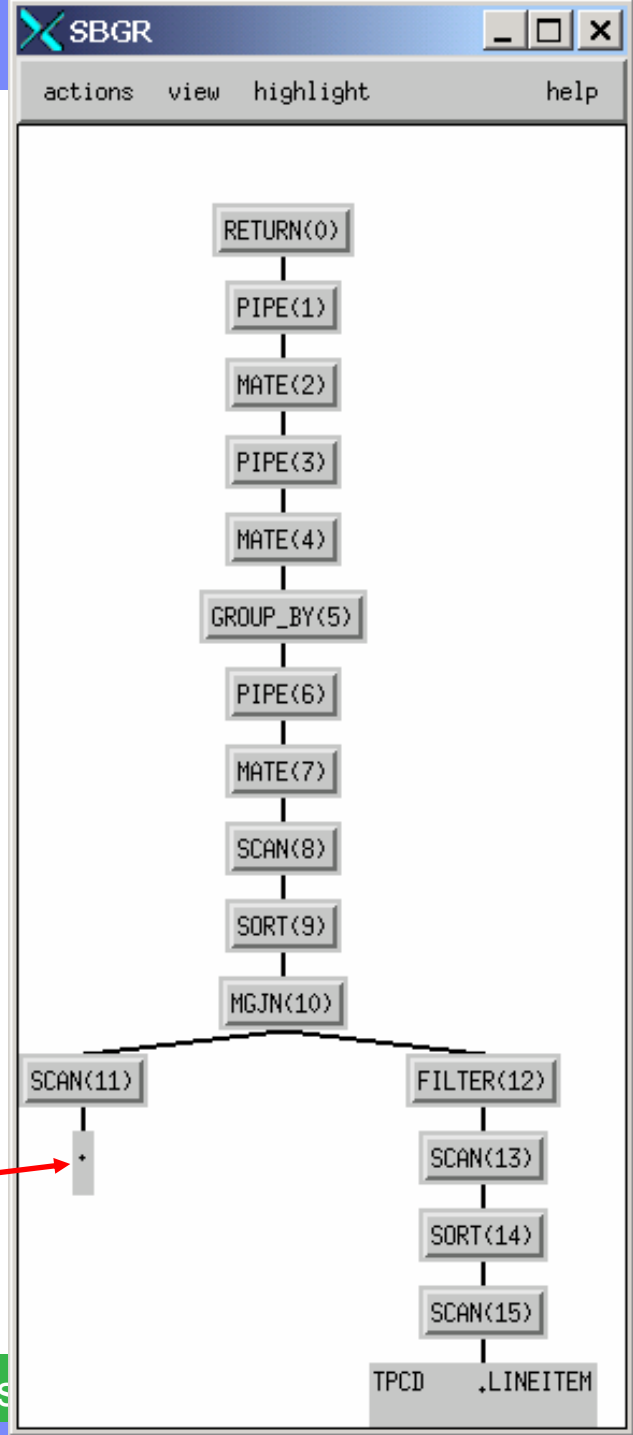
- Why wait till the query is finished to correct problem, as LEO does?
  - Can detect problems early!
  - Correct the plan dynamically,
    before we waste any more time!
  - May never execute this exact query again
- Long-running query won't notice re-optimization overhead
- Result: Plan more robust to optimizer mis-estimates!
- Complementary to LEO (different flavor)
  - POP fixes this plan
  - LEO fixes future plans



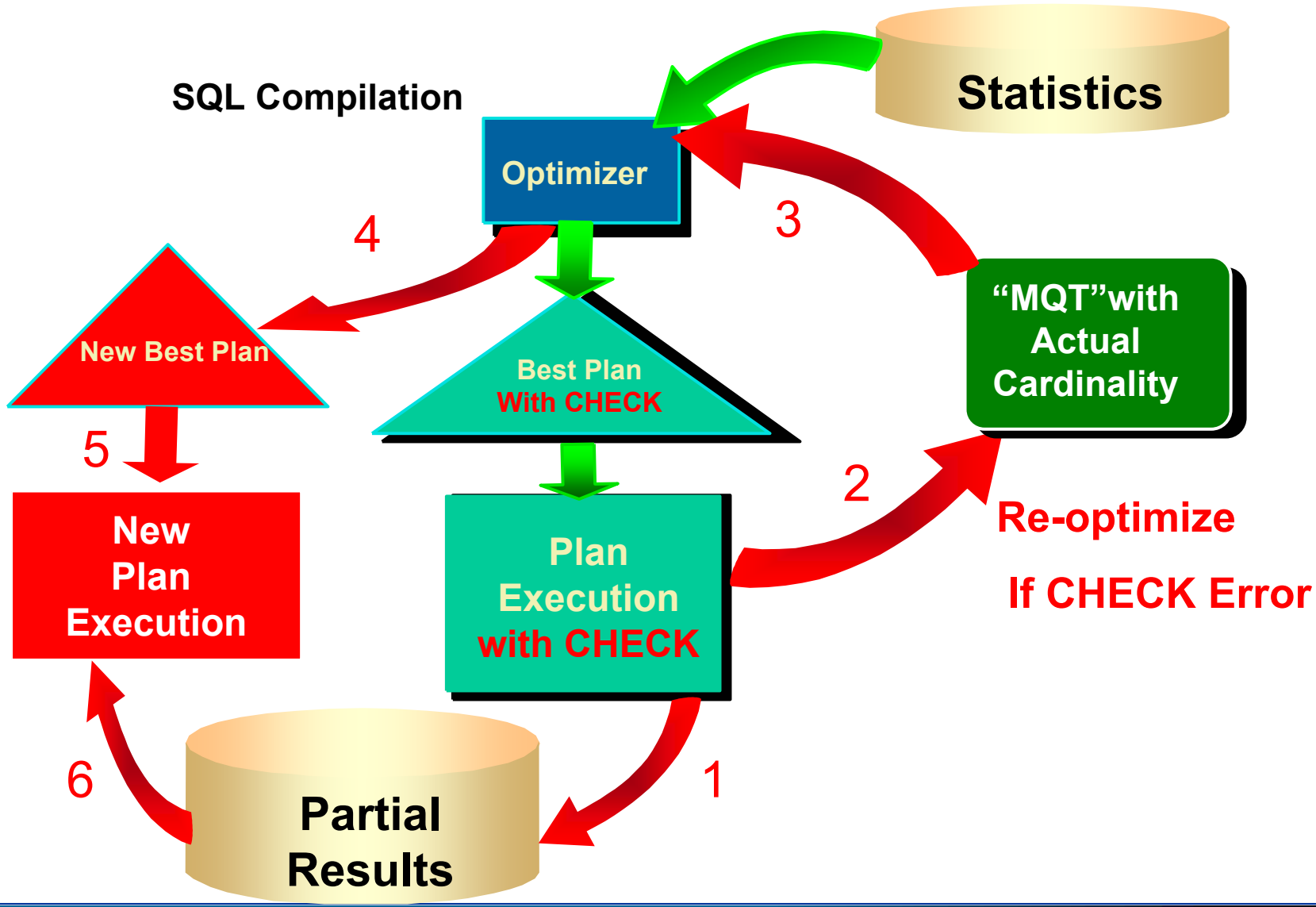
g

Progressive Optimization (POP)

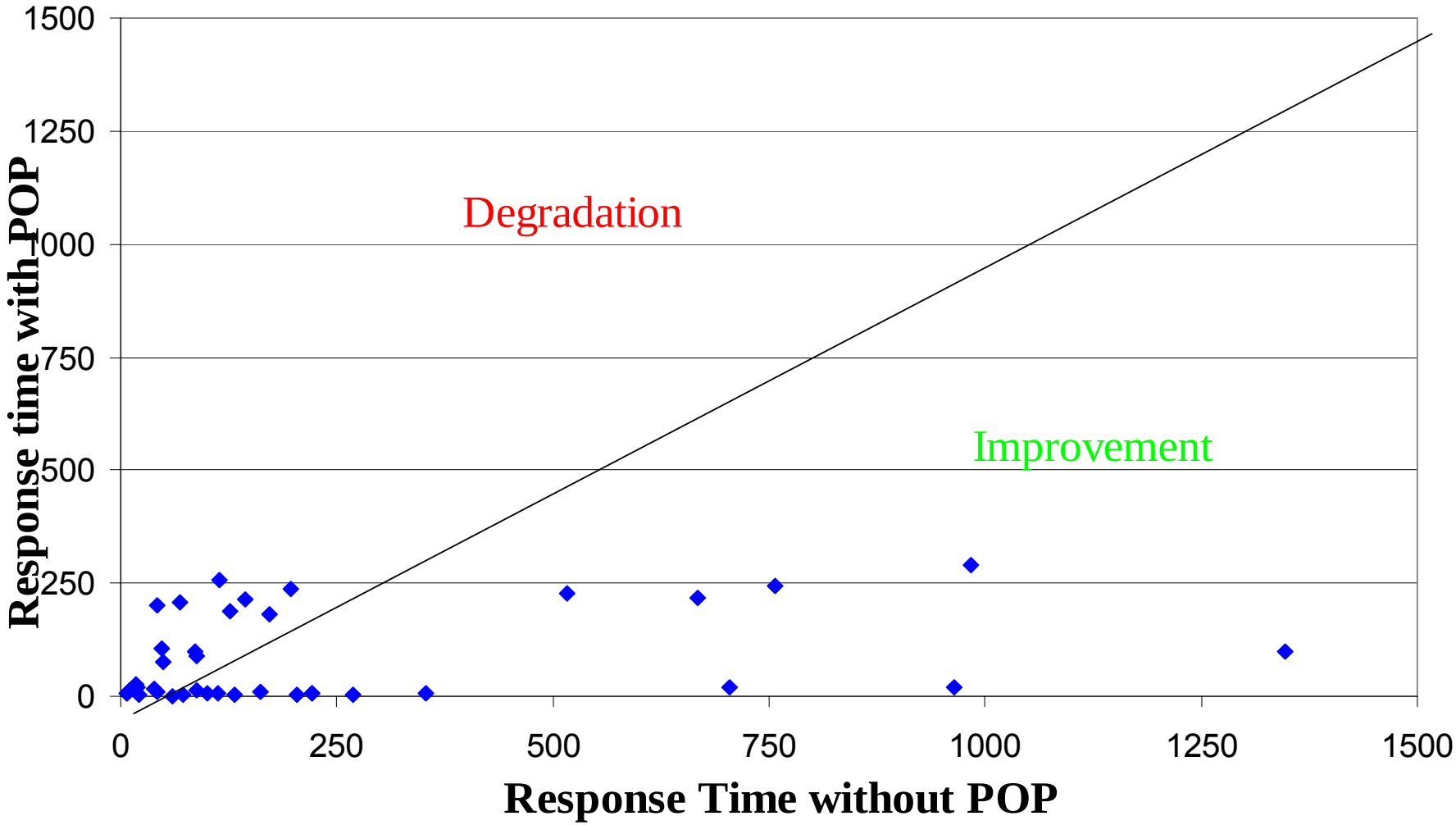
- **CHECKpoints for cardinality estimates at TEMP tables**
  - **Pre-computed validity range for this plan**
- **When check fails,**
  - **Treat partial results as MQTs**
  - **Replace estimated cardinality with actual for the MQTs**
  - **Re-optimize the currently running query**
- **Reuse results from partial execution**



POP: Re-Optimize Dynamically



Scatter Plot of Response Times (for Real World Database)



Publications

- General DB2 Autonomic Computing:
  - ✓ Usability and design considerations for an autonomic relational database management system, IBM Systems Journal 42, 4 (2003), pp. 568-581 (R. Telford, R.Horman, S. Lightstone, N. Markov, S. O'Connell, G. Lohman)
  - ✓ *SMART: Making DB2 (More) Autonomic*, **VLDB 2002** (Guy M. Lohman, Sam S. Lightstone)
  - ✓ *Toward Autonomic Computing with DB2 Universal Database*, in **ACM SIGMOD Record 31**,3 (Sept. 2002) (Sam S. Lightstone, Guy M. Lohman, Danny Zilio)
  - ✓ *A SMARTer DB2*, in **DB2 Magazine 7**,4 (Q4, 2002) (Guy M. Lohman, Bryan Smith, Sam S. Lightstone, Randy Horman, James Teng)
- Learning Optimizer (LEO):
  - ✓ Automated Statistics Collection in DB2 Stinger, **VLDB 2004**
  - ✓ CORDS: Automatic generation of correlation statistics in DB2 (Demo). I. Ilyas, V. Markl, P. J. Haas, P. G. Brown and A. Aboulnaga, **VLDB 2004**
  - ✓ Ihab F. Ilyas, Volker Markl, Peter J. Haas, Paul Brown, Ashraf Aboulnaga: CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. **ACM SIGMOD 2004**
  - ✓ Volker Markl, Vijayshankar Raman, David E. Simmen, Guy M. Lohman, Hamid Pirahesh: Robust Query Processing through Progressive Optimization. **ACM SIGMOD 2004**
  - ✓ Automatic relationship discovery in self-managing database systems. I. Ilyas, V. Markl, P. J. Haas, P. G. Brown and A. Aboulnaga. **Intl. Conf. Autonomic Computing (ICAC '04), 2004 (**poster)
  - ✓ Volker Markl, G. M. Lohman, and V. Raman. LEO: An Autonomic Query Optimizer for DB2. **IBM Systems Journal** Special Issue on Autonomic Computing, 1/2003
  - ✓ Volker Markl, Guy M. Lohman: Learning table access cardinalities with LEO. **ACM SIGMOD 2002** (demo): 61
  - ✓ Michael Stillger, Guy M. Lohman, Volker Markl, Mokhtar Kandil: LEO - DB2's LEarning Optimizer. **VLDB 2001**: 19-28
- Design Advisor:
  - ✓ DB2 Design Advisor: Integrated Automatic Physical Database Design, Daniel Zilio, Jun Rao, Sam Lightstone, Guy Lohman, Adam Storm, Christian Garcia-Arellano, Scott Fadden, **VLDB 2004**.
  - ✓ Recommending Materialized Views and Indexes with IBM's DB2 Design Advisor, **IEEE Intl. Conf. on Autonomic Computing (ICAC 2004)**
  - ✓ Trends in Automating Database Physical Design, **IEEE 2003 Workshop on Autonomic Computing Principles and Architectures**, Banff, Alberta, August 2003 (Danny Zilio, Sam Lightstone, and Guy Lohman).
  - ✓ Automating Physical Database Design in a Parallel Database: the DB2 Partitioning Advisor, Jun Rao, Chun Zhang, Guy Lohman, Nimrod Meggido, **ACM SIGMOD 2002**.

Publications (cont'd)

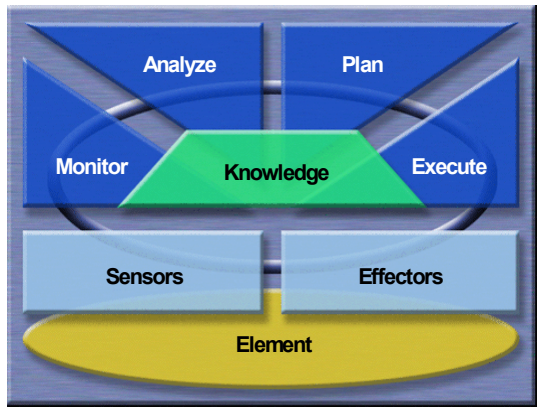
- Meta-Optimizer:
  - Estimating Compilation Time in a Query Optimizer, Ihab Ilyas, Jun Rao, Guy Lohman, Dengfeng Gao, Eileen Lin, **ACM SIGMOD 2003**.
- Sampling and "Bump Hunt":
  - A bi-level Bernoulli scheme for database sampling. P. J. Haas and C. Koenig. **SIGMOD 2004**
  - Data-stream sampling: basic techniques and results. P. J. Haas. In **Data Stream Management: Processing High Speed Data Streams**. M. Garofalakis, J. Gehrke, R. Rastogi (eds). Springer-Verlag, 2004 (to appear).
  - BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. P. G. Brown and P. J. Haas. **VLDB 2003**, 668--679.
  - The need for speed: speeding up DB2 using sampling. P. J. Haas. **IDUG Solutions Journal, 10**, 2003, 32--34.
- Other Optimizer:
  - Canonical Abstraction for Outerjoin Optimization, Jun Rao, Hamid Pirahesh, Calisto Zuzarte, ACM SIGMOD 2004.
  - Outerjoin and Antijoin Reordering Using Extended Eligibility Lists, Jun Rao, Bruce Lindsay, Guy Lohman, Hamid Pirahesh, David Simmen, **IEEE ICDE 2001**.
- Miscellaneous:
  - Roland Pieringer, Klaus Elhardt, Frank Ramsak, Volker Markl, Robert Fenk, Rudolf Bayer: Transbase: a Leading-edge ROLAP Engine Supporting Multidimensional Indexing and Hierarchy Clustering. **BTW 2003**: 648-667
  - Roland Pieringer, Klaus Elhardt, Frank Ramsak, Volker Markl, Robert Fenk, Robert Bayer, Nikos Karayannidis, Aris Tsois, Timos K. Sellis: Combining Hierarchy Encoding and Pre-Grouping: Intelligent Grouping in Star Join Processing. IEEE **ICDE 2003**: 329-340
  - Robert Fenk, Volker Markl, Rudolf Bayer: Interval Processing with the UB-Tree. **IDEAS 2002**: 12-22
  - Nikos Karayannidis, Aris Tsois, Timos K. Sellis, Roland Pieringer, Volker Markl, Frank Ramsak, Robert Fenk, Klaus Elhardt, Rudolf Bayer: Processing Star Queries on Hierarchically-Clustered Fact Tables. **VLDB 2002**: 730-741
  - Roland Pieringer, Volker Markl, Frank Ramsak, Rudolf Bayer: HINTA: A Linearization Algorithm for Physical Clustering of Complex OLAP Hierarchies. **DMDW 2001**: 11
  - Volker Markl, Frank Ramsak, Roland Pieringer, Robert Fenk, Klaus Elhardt, Rudolf Bayer: The Transbase Hypercube RDBMS: Multidimensional Indexing of Relational Tables. IEEE **ICDE Demo Sessions 2001**: 4-6
  - Martin Zirkel, Volker Markl, Rudolf Bayer: Exploitation of Pre-sortedness for Sorting in Query Processing: The TempTris-Algorithm for UB-Trees. **IDEAS 2001**: 155-166
  - Frank Ramsak, Volker Markl, Robert Fenk, Rudolf Bayer, Thomas Ruf: Interactive ROLAP on Large Datasets: A Case Study with UB-Trees. **IDEAS 2001**: 167-176
  - Efficient data reduction with EASE. H. Bronnimann, B. Chen, M. Dash, P. J. Haas, P. Scheuermann. **SIGKDD, 2003**, 59--68.
  - Efficient data reduction methods for on-line association rule discovery. H. Bronnimann, B. Chen, M. Dash, P. J. Haas, Y. Qiao, and P. Scheuermann. In t **Data Mining: Next Generation Challenges and Future Directions**. AAAI Press, 2004. In press.
  - A new two-phase sampling based algorithm for discovering association rules. B. Chen, P. J. Haas, and P. Scheuermann. **SIGKDD 2002**, 462--468.

Agenda

1. Motivation
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - Partition Advisor
   - LEarning Optimizer (LEO)
   - Progressive OPtimizer (POP)
3. **Self-Healing**
   - **Knowledge Bases for Problem Determination**
   - **Mining for Precursory Indicators**
4. Research Topics in Self-Managing Systems
5. Other Areas of Interest in My Group
6. Conclusions

Self-Healing Research: Knowledge Bases for Problem Determination

- Motivation:
  - ★ Self-diagnosing and self-healing crucial to self-managing systems
  - ★ **Hard problem** to determine cause of, and fix for, new problems!
- Build knowledge base from:
  - ★ Known problems (determined by humans)
  - ★ Knowledge mined from diagnostic & performance data
- Rationale:
  - ★ Many (~50%!) problems are re-occurrences of known problems!
  - ★ Easier than determining the root cause of a new problem
  - ★ Low-hanging fruit!
  - ★ Starting point for automation in Autonomic Manager (see diagram)
- Approach:
  - ★ Start simple with highly structured information:
    - ➢ Call Stacks (from crashes, hangs) parsed from text
    - ➢ Simple matching algorithms
  - ★ Expand to mine other diagnostic information
    - ➢ Structured (error codes, logs, traces, …)
    - ➢ Unstructured (keyword or textual descriptions, documentation,…)
  - ★ Add web query interface for users to submit queries
  - ★ Ultimately embed in Autonomic Manager



Autonomic Manager

("MAPE Loop")

Example Stack: Where's the Original Sin?

Common Error Handling Routines (Ignore)

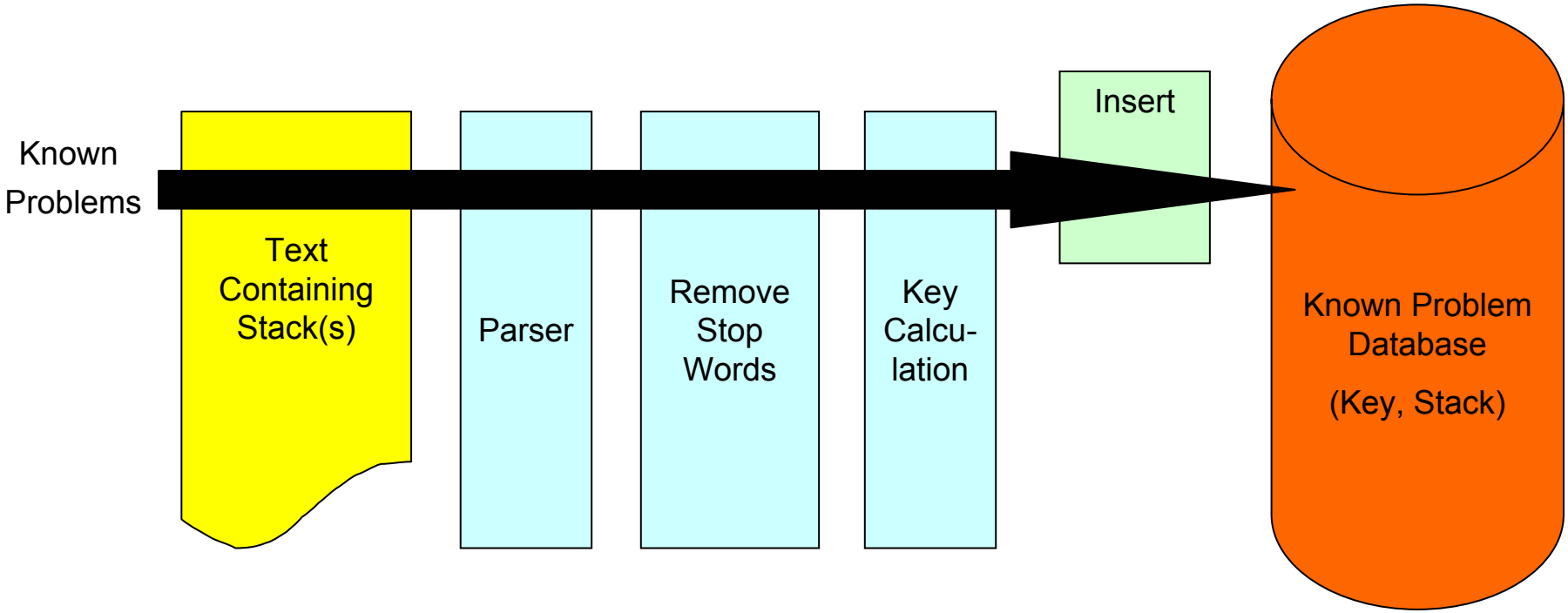
FAILURE!

Levels of Recursion Probably Not Relevant

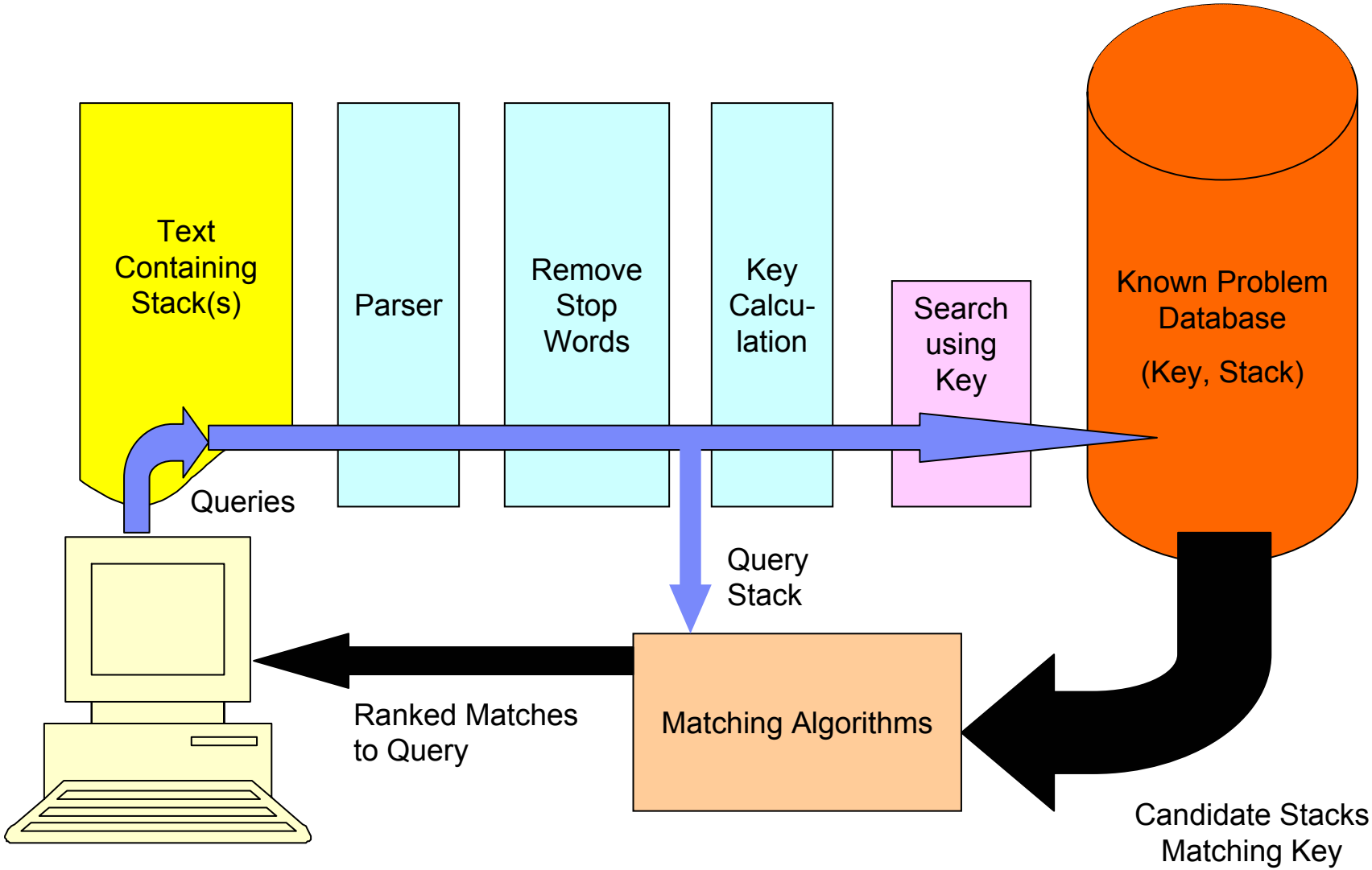
Entry-level Routines Definitely Not Relevant (Too Common)

```
6401261 *** Start stack traceback ***
0xD10ED244 sqloDumpEDU + 0x1C
0x200B852C sqldDumpContext__FP20sqle_agent_privatecbiN42PcPvT2 + 0x148
0xDE244A8C sqlrr_dump_ffdc__FP8sqlrr_cbiT2 + 0x520
0xD1220FC0 sqlzeDumpFFDC__FP20sqle_agent_privatecbUlP5sqlcai + 0x48
0xD1220BB0 sqlzeSqlCode__FP20sqle_agent_privatecbUiUlT3P5sqlcaiUsPc + 0x2D4
0xDF9C08EC sqlnn_erds__FiN41e + 0x174
0x2158F5D4 sqlno_ff_compute__FP13sqlno_globalsP19sqlno_ff_essentials + 0x628
0x2158FBBC sqlno_ff_or__FP13sqlno_globalsP19sqlno_ff_essentialsPf + 0x260
0x2158F384 sqlno_ff_compute__FP13sqlno_globalsP19sqlno_ff_essentials + 0x3D8
0x2158FBBC sqlno_ff_or__FP13sqlno_globalsP19sqlno_ff_essentialsPf + 0x260
0x2158F384 sqlno_ff_compute__FP13sqlno_globalsP19sqlno_ff_essentials + 0x3D8
0x2158FBBC sqlno_ff_or__FP13sqlno_globalsP19sqlno_ff_essentialsPf + 0x260
0x2158F384 sqlno_ff_compute__FP13sqlno_globalsP19sqlno_ff_essentials + 0x3D8
0x2172AF50 sqlno_ntup_ff_scan__FP13sqlno_globals + 0x10E0
0x2174D1EC sqlno_prep_phase__FP13sqlno_globalsP9sqlnq_qur + 0x1704
0x2174B29C sqlno_exe__FP9sqlnq_qur + 0x944
0xDFAA7C3C
    sqlnn_cmpl__FP20sqle_agent_privatecbP11sqlrrstrings17sqlnn_compileMod
eT3P14sqlrr_cmpl_envlT7PP9sqlnq_qur + 0x48E4
0xDFAA32CC
    sqlnn_cmpl__FP20sqle_agent_privatecbP11sqlrrstrings17sqlnn_compileMod
eT3P14sqlrr_cmpl_env + 0x68
0xDE514DD0
    sqlra_compile_var__FP8sqlrr_cbP14sqlra_cmpl_envPUciUsN54P16sqlra_cach
ed_varPiPUL + 0x1290
...
```

Architecture of Prototype: Population



Architecture of Prototype: Querying

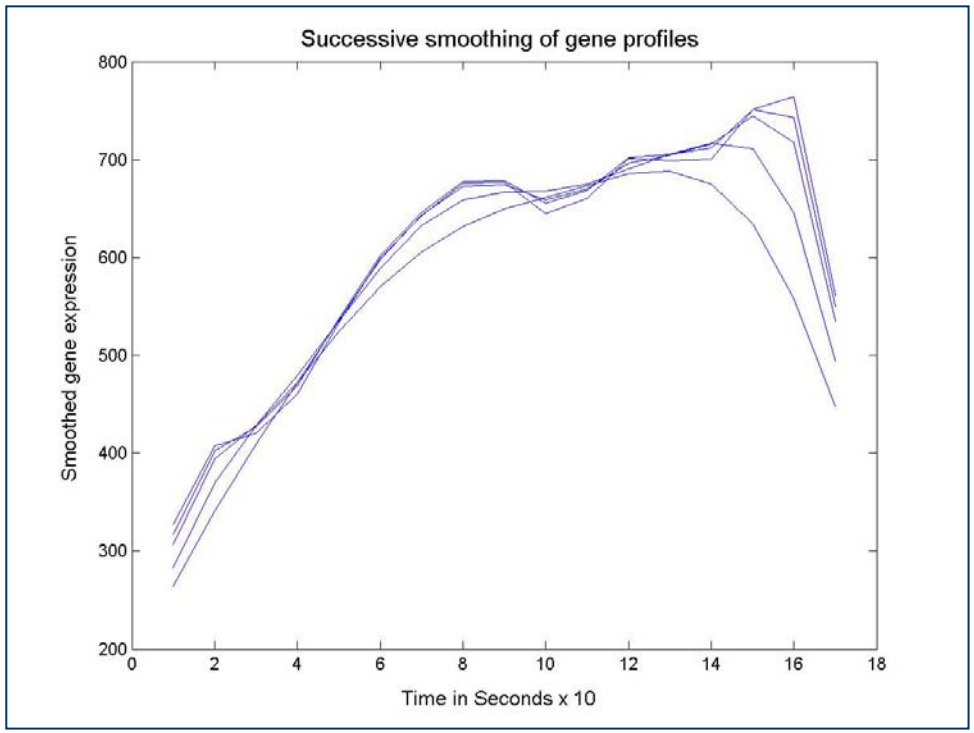


Prototype 2: New Problem Prediction

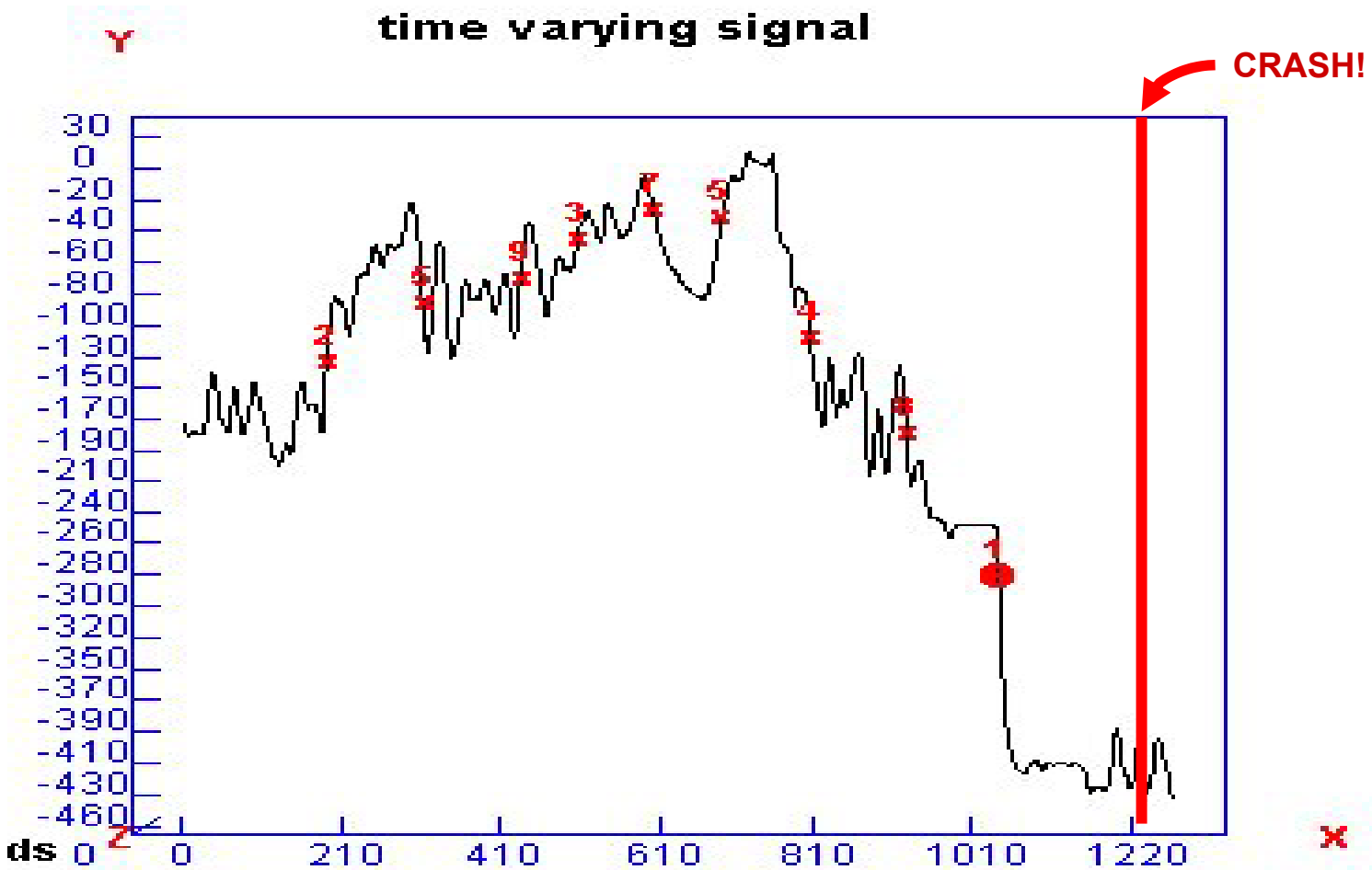
- Mine problem predictors from historical data
  - Historical data in Tivoli Data Warehouse
    - Numerous metrics of resource utilization
    - Known problem events recorded
  - Analyze by:
    - Traditional statistical analysis
      - Distribution fitting
      - Trend analysis
      - Spectral analysis (daily, weekly, monthly trends)
      - Correlation
    - Data mining
  - Develop real-time monitoring rules
  - Extend to time series patterns
    - Multi-variate correlation techniques
    - Exploit “salient events” in time series

Salient Change Points: Definition

- Inflection points are ***salient*** if they are preserved over multiple levels of smoothing.



Exploitation for Precursory Conditions



Agenda

1. Motivation
2. Self-Optimizing (DB2)
   - Design Advisor
     - Index Advisor
     - Partition Advisor
   - LEarning Optimizer (LEO)
   - Progressive OPtimizer (POP)
3. Self-Healing
   - Knowledge Bases for Problem Determination
   - Mining for Precursory Indicators
4. **Research Topics in Self-Managing Systems**
5. Other Areas of Interest in My Group
6. Conclusions

Research Topics / Issues (1 of 2)

- Capacity planning (modeling & estimation)
  - How model systems with limited specification?
  - How maintain model with evolving HW & SW?
- Installation
  - Dependency graph of prerequisite versions, configurations
- Database Design
  - Logical Design (application design, normalization)
  - Physical Design – how to decide:
    - Selection of indexes, materialized views, etc.
    - Data placement (clustering, partitioning, etc.)
    - Dynamic storage provisioning
- Performance tuning
  - How automate determination of poor performers?
  - How dynamically re-configure system in response to load changes?
- Maintenance – when / how to perform
  - Backups?
  - Reorganizations?
  - Statistics collection?
  - Upgrades?

Research Topics / Issues (2 of 2)

- Self-Healing
  - How much monitoring data to collect?
  - How do you know if your system is “firing on all cylinders”?
  - How do you isolate problems from noise of diagnostics?
  - How do you correlate logs from components on different machines w/ diff. clocks?
  - How do you isolate root cause from cascading error messages?
  - Fuzzy searching of symptom databases
  - How do you automatically generate diagnostics to resolve ambiguous problems?
  - How do you model and determine the cause & repair for problems never before seen?
  - How do you determine the best fix for a problem, even if the cause is known?
  - How do you build repair rules automatically from past successes & failures?
- System Control
  - Scheduling & prioritization of tasks
  - How do you resolve conflicting rules & priorities?
  - How do you make progress on maintenance without impacting production?
  - How do you avoid instability and “thrashing” (control theory)?
  - How much monitoring is enough to resolve problems but not impact production?
  - How do you learn from past successes & failures?

Other Areas of Interest in My Group

- Optimizing XQuery
- Automatically deriving cost models
- Outer join optimizations
- Sampling databases for fun and profit
- Deriving semantic information from databases via sampling
  - "Bump Hunt" – algebraic relationships
  - CORDS – CORrelation Detection by Sampling

Conclusions

- Many Self-Managing features already delivered to DB2
- Many exciting Self-Managing projects in the pipeline!
- More I couldn't talk about today
- **Exploit** and **improve** DB2's Optimizer
- Next-generation query optimization technology
- Intend to:
  - ☑ Extend our lead in Optimization technology
  - ☑ Use it to optimize the system, not just queries!
  - ☑ Make DB2 self-managing -- Free the DBAs!

Appendix

Sampling in SQL Queries

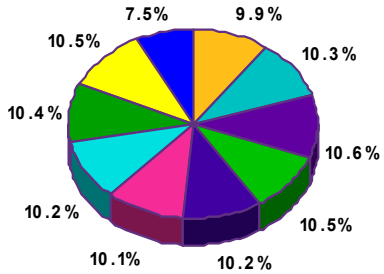
(Peter Haas, Paul Brown, Ashutosh Singh, Mike Winer (TOR))

- **The Problem:**
  - Data warehouse
  - Business Intelligence/OLAP queries
  - Database volumes of 100's of terabytes
  - User on a "fishing expedition"
  - Aggregation
  - Imprecise results OK
  - Provide error estimate of sample
- **Solution:** ***Sampling!***
  - Process less data
  - Huge performance improvement (orders of magnitude)
  - Results "close" to complete answer
- Also useful for RUNSTATS!

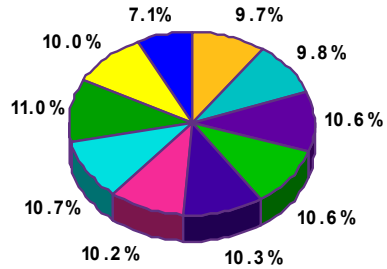
Results: Can you tell the difference?

- Processing less data => huge performance improvement
- Approximate answers often suffice

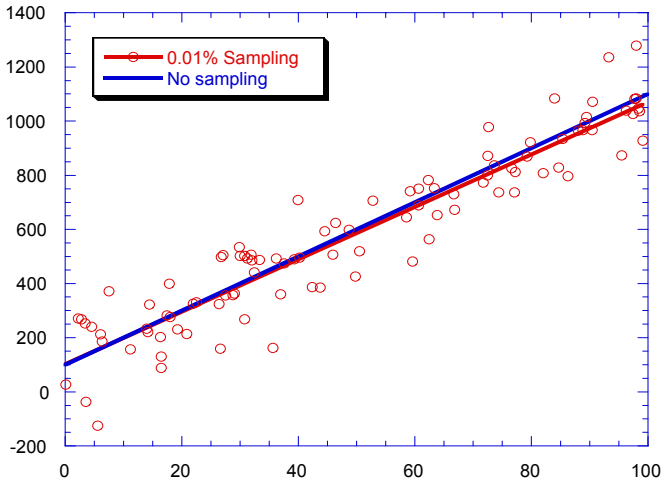
No sampling (25 min.)



1% sampling (2 sec.)

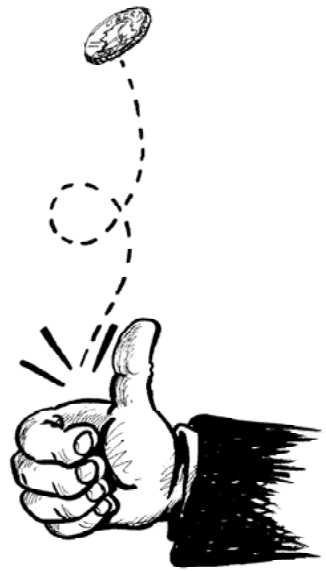


Line Fit



Pure Row-Level Bernoulli Sampling

- Bernoulli ("coin-flipping") sampling:
  - Flip a (weighted) coin for each row
  - Select or reject based on a user-specified probability
- Why Bernoulli Sampling?
  - simple
  - interacts well with SQL (treat sampling clause as a predicate)
  - Well-suited to parallelism
- Example in DB2 for Unix and Windows (< V8):
  WHERE RAND() < 0.01
- Shortcoming: saves mostly CPU, not I/O!
  - If any index on sampled table exists, must touch
    - All leaf pages of index
    - Almost 1 data page per row retrieved
  - Otherwise, every row (and page) touched!



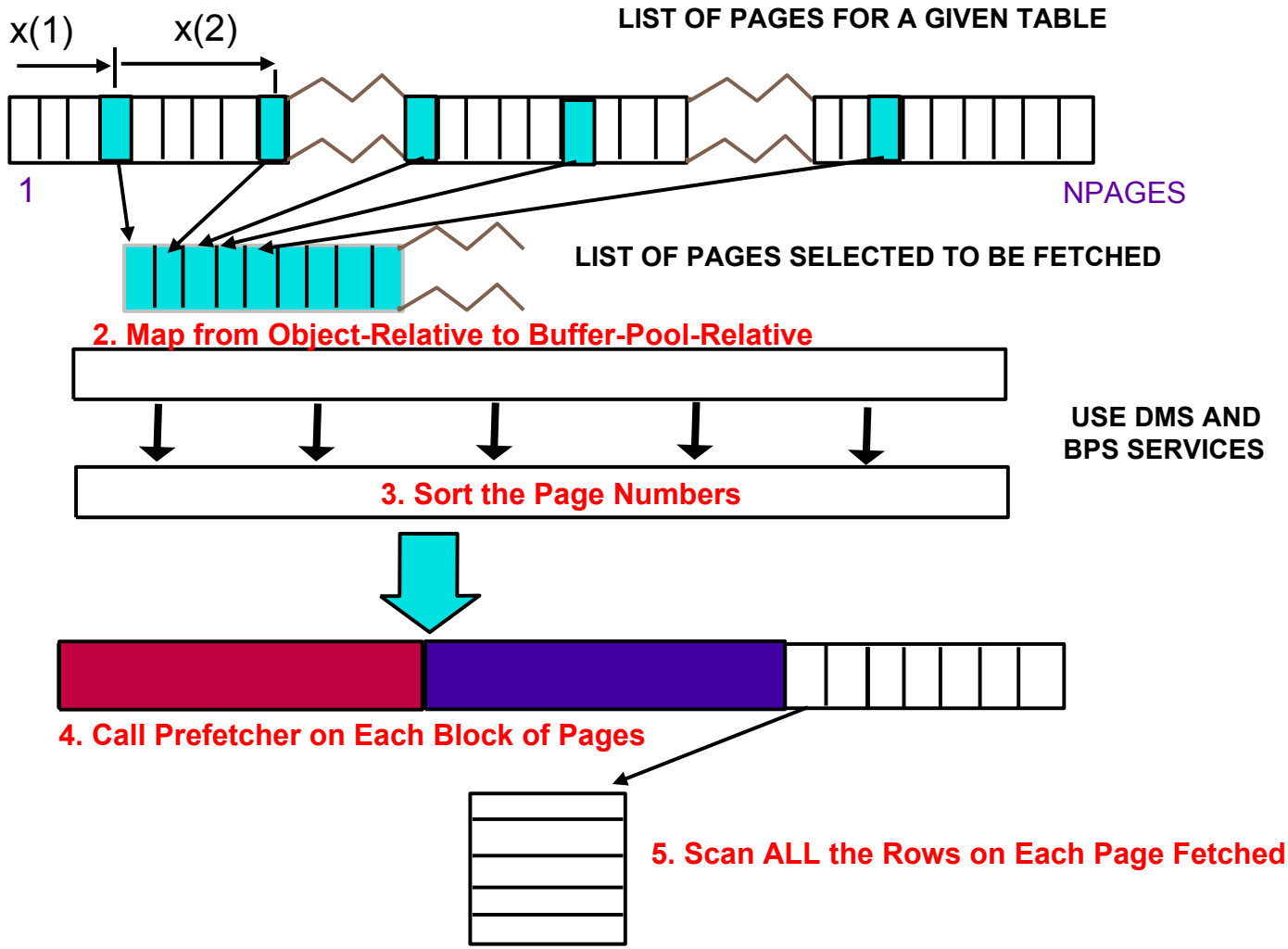
More Efficient Approach: Page-Level Sampling

- Sample pages first
  - Select each page with probability P (save I/O cost)
  - Use all rows on that page
- Advantage: dramatic I/O savings!
  - More efficient use of rows on pages touched
  - Quick algorithms for generating page lists
  - Know where to find those pages
  - Hence, can prefetch

What's Going On…

Page-Level Sampling: What's Going On?

1. Use Random function to determine X(i)



EXAMPLE -- A Simple Sampling Query

- **Query:**

  SELECT sum(ti.qty * ti.price)/.01 AS amount,
  FROM stars.transitem1 AS ti TABLESAMPLE BERNOULLI(1)

- **Syntax**
  - TABLESAMPLE clause
    - added to table in the FROM clause
    - specifies sampling rate,
      - as a percent
      - can be fractional (e.g. 0.05)
  - Draft ISO standard (negotiated with Oracle, Teradata)
- **Semantics**
  - Rows retrieved not exactly P percent (# of rows may vary)
  - Values of SUM and COUNT are for rows retrieved
  - Need to divide by sampling rate to extrapolate these to entire database (in this query)

There's Always a Fly in the Ointment!

- **Problem:** What if data is clustered on sampled columns?
  - Each page contains little variation in values
  - EXAMPLE:
    - SELECT AVG(salary) FROM EMP TABLESAMPLE (0.1)
    - suppose EMP is clustered by salary
  - Sampling gets a "myopic" view from the pages it samples
  - Can get wrong (biased) results
  - Thinks the error of the sample is better than it really is

- **Solution:** have to sample more pages, and fewer rows per page
  - “Bi-Level Bernoulli Sampling”
  - How determine the optimal tradeoff between
    - Accuracy
    - Efficiency
  - How automatically detect and correct for this?

Sampling Support in DB2 for LUW: Status

- Prior to V8.1:
  - **SELECT * FROM T WHERE RAND() < 0.01**
  - **Requires full scan of table (no I/O savings)**
- Released in DB2 V8.1 FP2 (Spring 2003):
  - **SELECT * FROM T TABLESAMPLE SYSTEM(1)**
  - **SYSTEM = page-level Bernoulli sampling (best performance)**
  - **Supports index-aware BERNOULLI method (best accuracy)**
  - **Supports REPEATABLE clause**
  - **Works with SMP/MPP parallelism, MDC, etc. (but not Federated)**
  - **Draft ISO standard**
  - **Currently available to certain customers (Home Depot)**
  - **Customer education: IDUG, DB2 Tech. Conf.**
- Future Work
  - **SAMPLE UNIT clause for estimation of Standard Error**
  - **Bi-level Bernoulli Sampling to deal with bias due to clustering**
  - **Statistics to automatically choose best parameters**
  - **Bridge to mining & analysis applications (e.g. SAS)**

Meta-Optimizer

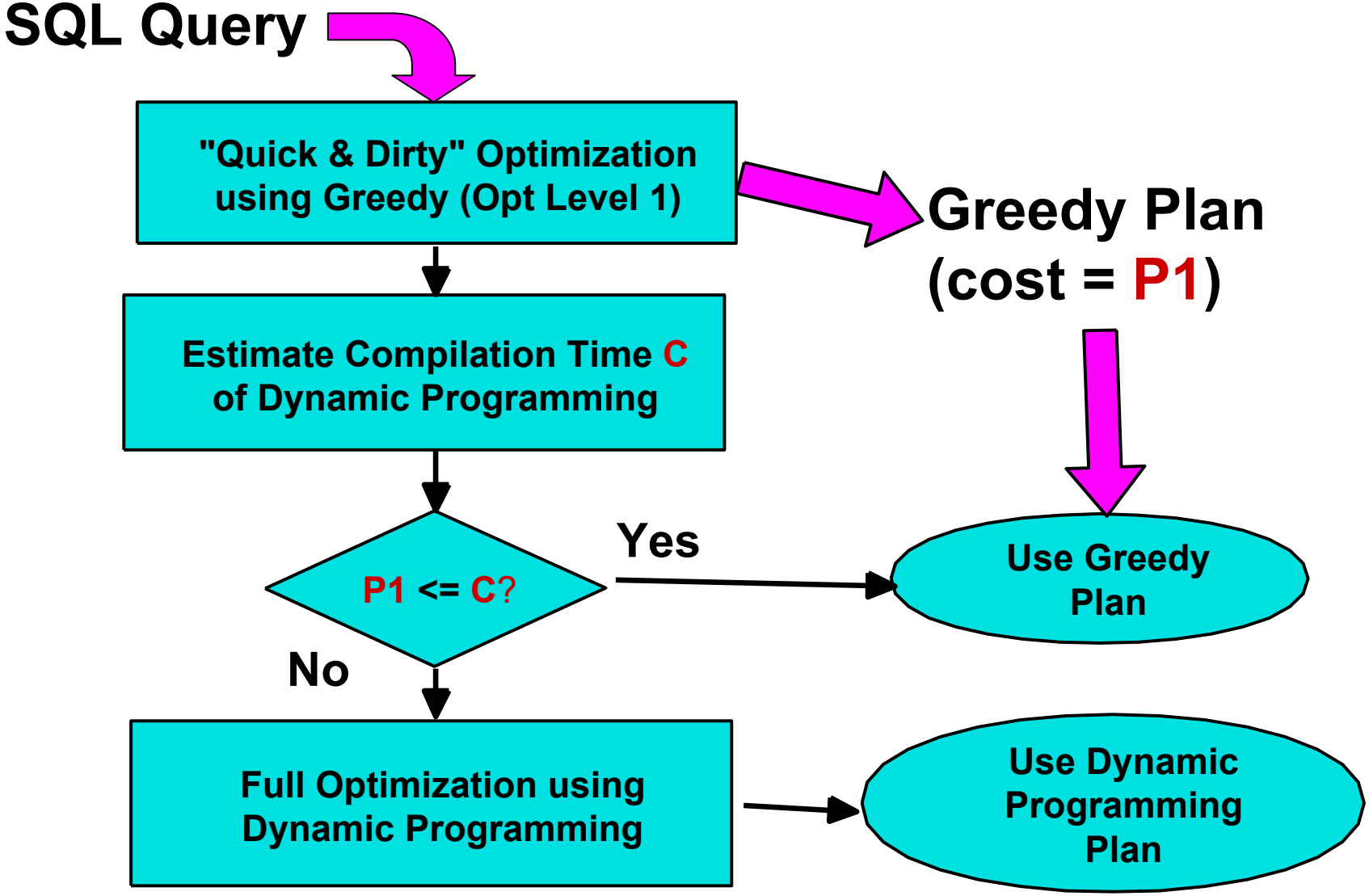
(Jun Rao, Ihab Ilyas)

- Goals
  - Automatically decide best optimization level (degree of optimization)
  - Improve over current heuristic approach
- "Pilot Pass" Approach
  - Get "quick & dirty" plan using Greedy (level 1)
  - Estimate time to optimize at higher levels of optimization
  - Decide if extra optimization is worth the compilation time
- Challenge: Characterizing compilation time as a function of
  - Number of tables
  - "Shape" of query graph's join predicates -- star vs. chain
  - Multiple join predicates between same tables
  - Number of "interesting orders"
  - Number of "interesting partitions" (in EEE)
  - Others?

- Reference: "Estimating Compilation Time of a Query Optimizer", SIGMOD 2003, Ihab Ilyas, Jun Rao, Guy Lohman, et al.

Meta-Optimizer

(Jun Rao, Ihab Ilyas)



What do we want to learn?

1. Base table cardinalities
2. Filter factors for:
   A. Local predicates
   B. Joins
3. Detect correlations
4. Bad & good estimates
5. Subquery estimates
6. ANY operation!