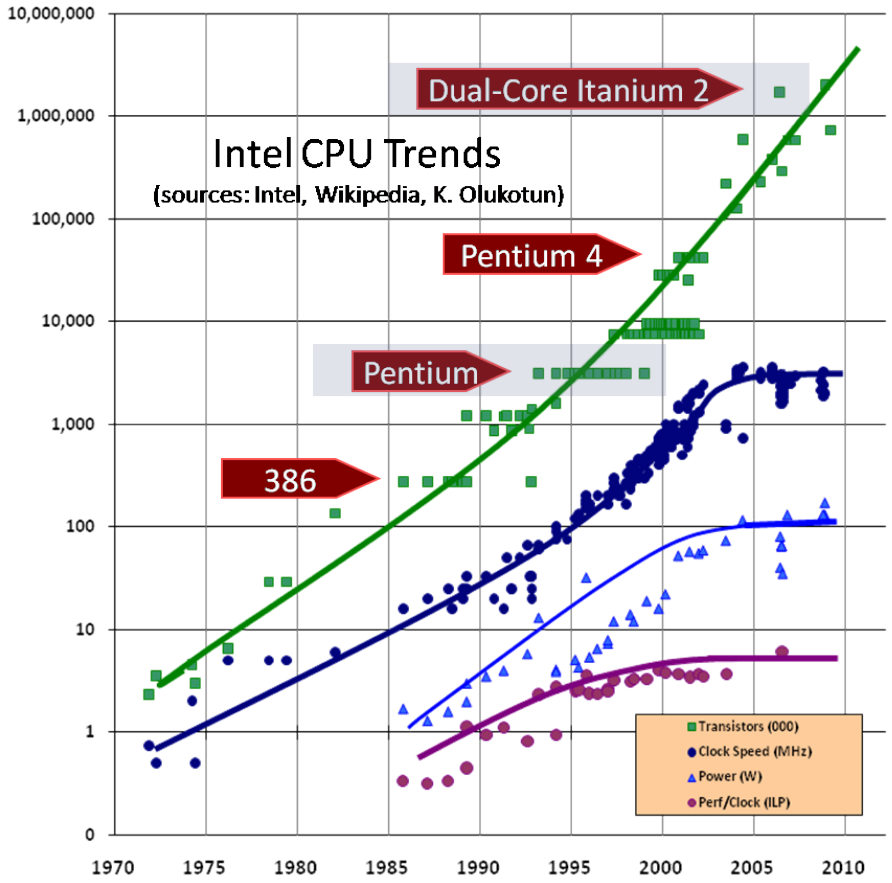
Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Parallel Programming for In-Memory Databases

Martin Grund,
Johannes Wust
Alexander Zeier

“The Free Lunch Is Over”



http://www.gotw.ca/publications/concurrency-ddj.htm

- Number of transistors per CPU increases
- Clock frequency stalls

Challenge

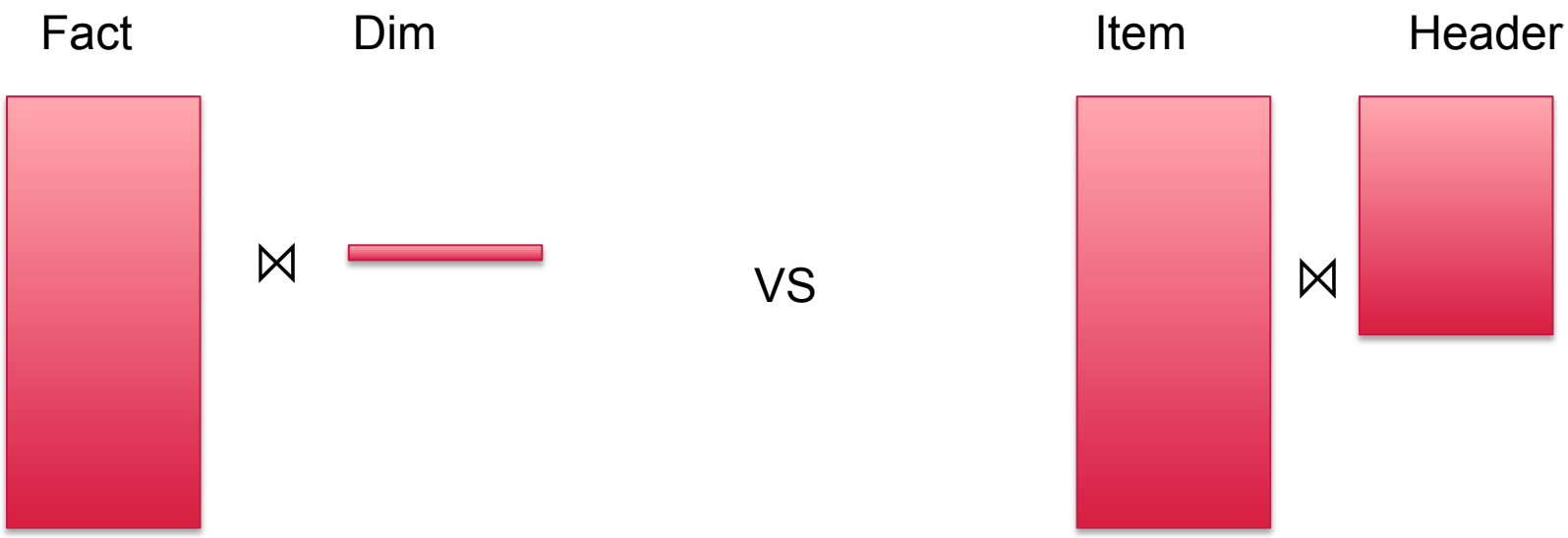
- Modern enterprise database world is split into two worlds:
  - Analytical processing
  - Transactional processing
  - (Stream processing)

Moving data from transactional systems to analytical system is a bottleneck and generate a maintenance overhead

- In-Memory database provide the potential to unlock the unification of the analytical and transactional world
  - High update rate
  - High scan speed

Challenges Contd.

- Goal: Analytical operations on transactional data
- Problem: Join cardinality



Today's database systems are not optimized for analytical queries on transactional data

Lecture Goal

- Given a database schema and a set of queries, how can one optimize the queries to allow optimal execution?
  - `SELECT SUM(DMBTR), KUNNR FROM BSEG, BKPF WHERE BSEG.BELNR = BKPF.BELNR GROUP BY KUNNR`
  - Which join algorithm? (Nested Loop, Hash, Radix, Sort-Merge)
  - What kind of aggregation?
- Given a data parallel execution, what is the optimal way to implement the plan operators?

Lecture Project a.k.a. Programming Challenge

- Team together with max 3 students to build the best aggregation / join operators for HYRISE
  - Main memory storage engine written in C++
- You will receive a set of sample data and a sample workload
  - Workload as SQL -> you translate to our intermediate format
- You optimize the system
- You implement the plan operations
- We test your implementation on the biggest machine we can get (most likely 64 cores, 2 TB RAM)

  - **The fastest implementation wins the challenge!**

Contest Rules

- You have to implement your query using HYRISE plan operations
- Input / Output are tables
- No intra-plan operation parallelization
- No fully materialized aggregates (no explicit caching)
- Your code is BSD licensed

- What happens if
  - *… I modified the storage implementation and everything runs faster?* – We will merge the code into the master and make it accessible to everyone. Feel free to contribute!
  - *… I have ideas for other parts of the implementation?* – See above!

Final Presentation

- Why a final presentation?
  - Explain your starting point and how you evolved your implementation
  - Show your ideas
  - Present your implementation, explain your implementations properties (“high-level” documentation)

Lecture Schedule

Part I

- Introduction to Enterprise Applications
- Main Memory and Modern Hardware
- Parallel Programming (Intel Open Course)
- HYRISE Deep Dive

Part II

- Challenge Implementation
- Mandatory Biweekly consultation

Part III

- Final Presentation
- Contest Evaluation

Administrative info at a glance

- 6 Leistungspunkte
- Teams of max 3 people, in total max 4 teams
- ~4 weeks lecture with exercises: Thursdays, 9:15 – 10:45, v2.16
- 8 weeks implementation: Biweekly consultations per team
- Final presentation to demonstrate approach and results:
- Date 02/09/2012
- Contact:
  - Martin Grund (v2.05)
  - Johannes Wust (v2.05)

Lecture Grading

- 30% implementation quality (documentation, test coverage, usability)
- 40% problem solving (algorithms, partitioning, etc.)
- 30% final presentation

- Best team will not necessarily get the best grade!

Lecture Literature

- See Mendeley Group - http://www.mendeley.com/groups/1571273/parallel-programming-for-imdb/
- Think about your programming and language skills!
  - http://www.slideshare.net/olvemaudal/deep-c