Applying In-Memory Technology to Genome Data Analysis

Cindy Fähnrich
Hasso Plattner Institute
GLOBAL HEALTH ’14 Tutorial
Hasso Plattner Institute
Key Facts

- Founded as a public-private partnership in 1998 in Potsdam near Berlin, Germany
- Institute belongs to the University of Potsdam
- Ranked 1st in CHE since 2009
- 500 B.Sc. and M.Sc. students
- 10 professors, 150 PhD students
- Course of study: IT Systems Engineering
Hasso Plattner Institute Programs

- Full university curriculum
- Bachelor (6 semesters)
- Master (4 semesters)
- Orthogonal Activities:
  - E-Health Consortium
  - School of Design Thinking
  - Research School

Dr. Schapranow, HPI, Aug 12, 2014
Prof. Dr. h.c. Hasso Plattner

- Research focuses on the technical aspects of enterprise software and design of complex applications
  - In-Memory Data Management for Enterprise Applications
  - Enterprise Application Programming Model
  - Scientific Data Management
  - Human-Centered Software Design and Engineering

- Industry cooperations, e.g. SAP, Siemens, Audi, and EADS
- Research cooperations, e.g. Stanford, MIT, and Berkeley
Agenda

1. **Introduction to In-Memory Technology**
2. Introduction to Genome Data Analysis
3. Combining In-Memory Technology with Genome Data Analysis
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - IMDB Analysis Features for Applications
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combined column and row store</td>
<td>Insert only for time travel</td>
</tr>
<tr>
<td>Active/passive data store</td>
<td>Dynamic multi-threading within nodes</td>
</tr>
<tr>
<td>No aggregate tables</td>
<td>On-the-fly extensibility</td>
</tr>
<tr>
<td>On-the-fly extensibility</td>
<td>Map reduce</td>
</tr>
<tr>
<td>Minimal projections</td>
<td>Bulk load</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Analytics on historical data</td>
</tr>
<tr>
<td>Single and multi-tenancy</td>
<td>Object to relational mapping</td>
</tr>
<tr>
<td>Any attribute as index</td>
<td>Reduction of layers</td>
</tr>
<tr>
<td>Lightweight Compression</td>
<td>Multi-core/parallelization</td>
</tr>
<tr>
<td>SQL interface on columns &amp; rows</td>
<td>Lightweight Compression</td>
</tr>
<tr>
<td>Text Retrieval and Extraction</td>
<td>No disk</td>
</tr>
</tbody>
</table>
Combined Column and Row Store

- Row stores are designed for operative workload, e.g.
  - Create and maintain meta data for tests
  - Access a complete record of a trial or test series

- Column stores are designed for analytical work, e.g.
  - Evaluate the number of positive test results
  - Identification of correlations or test candidates

- In-Memory approach: Combination of both stores
  - Increased performance for analytical work
  - Operative performance remains interactively
Traditional databases allow four data operations: INSERT, SELECT, DELETE, UPDATE

- DELETE and UPDATE are destructive since original data is no longer available

- Insert-only requires only first two to store a complete history (bookkeeping systems)

- Insert-only enables time travelling, e.g. to
  - Trace changes and reconstruct decisions
  - Document complete history of changes, therapies, etc.
  - Enable statistical observations
Main memory access is the new bottleneck

Lightweight compression can reduce this bottleneck, i.e.

- Improved usage of data bus capacity
- Work directly on compressed data

```
<table>
<thead>
<tr>
<th>recID</th>
<th>fname</th>
<th></th>
<th>valueID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>John</td>
<td></td>
<td>23</td>
<td>John</td>
</tr>
<tr>
<td>40</td>
<td>Mary</td>
<td></td>
<td>24</td>
<td>Mary</td>
</tr>
<tr>
<td>41</td>
<td>Jane</td>
<td></td>
<td>25</td>
<td>Jane</td>
</tr>
<tr>
<td>42</td>
<td>John</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>position</th>
<th>valueID</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>40</td>
<td>24</td>
</tr>
<tr>
<td>41</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>fname</th>
<th>valueID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Dictionary for “fname”
IMDB paradigm: data stored at highest possible level of granularity

Contrast to current practice of business data centers

- Store on level of granularity required by application
- Multiple applications use same data but require different granularity

→ High data redundancy and maintenance efforts

IMDB computes aggregates from source data on the

→ Dramatical complexity decrease, easier maintenance
Partitioning

- **Horizontal Partitioning**
  - Cut long tables into shorter segments
  - E.g. to group samples with same relevance

- **Vertical Partitioning**
  - Split off columns to individual resources
  - E.g. to separate personalized data from experiment data

- Partitioning is the basis for
  - Parallel execution of database queries
  - Implementation of data aging and data retention management
Modern server systems consist of $x$ CPUs, e.g.
- Each CPU consists of $y$ CPU cores, e.g. 8
- Consider each of the $x*y$ CPU core as individual workers
- Each worker can perform one task at the same time in parallel

Full table scan of database table with 1M entries results in $1/x*1/y$ search time when traversing in parallel
- Reduced response time
- No need for pre-aggregated totals and redundant data
- Improved usage of hardware
- Instant analysis of data
Active and Passive Data Store

- Active data are accessed frequently & updates are expected, e.g.
  - Most recent experiment results, e.g. last two weeks
  - Samples that have not been processed yet

- Passive data are used for analytical & statistical purposes, e.g.
  - Samples that were processed 5 years ago
  - Meta data about seeds that are not longer produced

- Moving passive data on slower storages
  - Reduces main memory demands
  - Improves performance for active data
Layers are introduced to abstract from complexity

Each layer offers complete functionality, e.g. meta data of samples

Less layer result in
- Less code to maintain
- More specific code
- Reduced resource demands
- Improves performance of applications due to eliminating obsolete processing steps
In-Memory Databases – History

- Original use case in 2006: Enterprise software
  - Combining operational and analytical data into one database
  - Enable real-time analysis on latest data

- Big data context: Business and accounting data, customer records, sales orders, invoices, ...

- Started 2009 to use in-memory technology in the context of life sciences

- Big data context: Genomic/biological data, prescriptions, patient and cancer records, clinical information systems, medical publications, ...
Agenda

1. Introduction to In-Memory Technology
2. **Introduction to Genome Data Analysis**
3. Combining In-Memory Technology with Genome Data Analysis
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - IMDB Analysis Features for Applications
“Personalized medicine aims at treating patients specifically based on their individual dispositions, e.g. genetic or environmental factors”


- Conventional cancer therapies often fail
  - One therapy does NOT fit all
  - Relation between genetic mutations and disease not considered/understood
  → Analyze genetic profile of a patient to define customized therapies

- Challenge: Complex and time-consuming data processing tasks (analysis of one patient takes up to weeks)...
Base Sequencing

- Deriving DNA in digital format from sample via imaging procedures
- Output are unordered DNA snippets (=reads)
- High error rate → Sequencing at multiple coverage
Genome Data Analysis

**DNA Sample**
- Base Sequencing: chemical
- Read Alignment: computational
- Variant Calling: computational
- Data Annotation: computational/manual

**Analysis Results**

**Alignment**
- Reconstruct genome by reassembling all reads
- Pattern-matching vs. similarity search
- Matching strictness vs. runtime performance
Variant Calling

- Detecting genetic variants in the sample genome
- Comparison to a reference
- Incorporating error probability of data
Genome Data Analysis

**Data Annotation**

- Find out impact of detected genetic variants on organism
- Connect known information, e.g. from studies/research papers, to genetic variants
- Gain new research insights, e.g. relations between genes and diseases, for personalized medicine
Process requires intermediate steps to improve data quality
- Deduplication: Exclude duplicate reads from analysis
- Local Realignment: Reduce false positives caused by Indels
- Base Quality Score Recalibration (BQSR): Adjust error probabilities of bases
- Variant Quality Score Recalibration (VQSR): Adjust variant probabilities
Insertions or Deletions (InDels) in reads can “trick” alignment algorithms into misaligning reads and introducing false positive Single Nucleotide Polymorphisms (SNPs).

**Reference:** TTTTTTTCGAT

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGA</td>
<td>CGA</td>
<td>CGA</td>
<td>CGA</td>
<td>CGA</td>
<td></td>
</tr>
</tbody>
</table>

C, G, A are recognized as SNPs, although these reads seem to contain a hidden deletion of a T!

**Reference:** TTTTTTTTCGAT

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local realignment introduces a Deletion for these reads, so the falsely detected SNPs disappear!
Genome Data Analysis – Alignment and Variant Calling Refined (2/3)

- Process requires intermediate steps to prepare data for faster processing
- More complicated when splitting up Alignment and Variant Calling …
Alignment and Variant Calling

- Single tasks are triggered manually or in scripts invoking tools via command line:

  ```bash
  bwa aln ref.fa sample.fastq | bwa samse ref.fa - sample.fastq | samtools view -Su - | samtools sort ...
  ```

- Effective parallelization?
- Error handling?
- Distribution to a cluster?

Data Annotation

- Mostly manual analysis, e.g. via keyword search in portals on the web
- Efficient analysis of data from a patient/cohort?
Agenda

1. Introduction to In-Memory Technology
2. Introduction to Genome Data Analysis
3. Combining In-Memory Technology with Genome Data Analysis
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - IMDB Analysis Features for Applications
Pipeline Modeling – How to Set Up a Pipeline

- Analysis pipeline is constructed from combining tools for the different analysis steps
  - Currently manual work via command line piping/scripts
  - Hard to understand/document/maintain

- Objective: Model the analysis pipeline with ...
  - ... a graphical representation that is ...
  - ... easy to understand and adapt

- Prerequisite: Graphical notation with standardized, machine readable representation
Pipeline Modeling – BPMN 2.0

- Business Process Model and Notation (BPMN) 2.0
- Functional modeling of business processes and workflows
- Intended for both business and technical users → intuitive modeling
- XPDL available as XML standard for representing BPMN
BPMN 2.0 – Basic Notation Overview

- **Start Event**
- **Task**
- **Parallel Gateways**
- **Exclusive Gateways**
- **Collapsed Subtask**
- **Task A**
- **Subtask B**
- **Task C**
- **Task D**
- **Task E**
- **End Event**
- **Multiple Task Instances Executed in Parallel**
- **Annotated Data Objects**
- **Data Object**
Pipeline Modeling with BPMN

- Model and adapt your models in your tool of choice

- Only using a subset of BPMN, adapted with own constructs:
  - Modular structure
  - Degree of parallelization
  - Parameters
  - Variables
Pipeline Modeling – Modular Structure

- Pipeline models can be nested hierarchically
- Reuse existing pipeline components, e.g. for alignment
- Make pipeline flexible regarding the tools used
Pipeline Modeling – Degree of Parallelization

- Execute parts of the pipeline in parallel
- Configure the explicit amount of parallel instances
Pipeline Modeling – Parameter

- Some tasks require parameters to be executed
  - Reference genome
  - Thread size
  - Number of parallel instances

- Annotation of tasks with explicit parameters via data objects
Pipeline Modeling – Variables

- Some parameter values cannot be specified at design time
  - Number of parallel instances
  - Reference genome

- Annotate tasks with variables that are set at runtime
Pipeline Modeling –
Creating the Final Analysis Pipeline

- Specify all subprocess models, parameters, variables

- Import all models in XPDL format into database

- Database entry of a pipeline model consists of
  - Name
  - Model ID
  - List of subprocess IDs
  - List of parameters and variables
Agenda

1. Introduction to In-Memory Technology
2. Introduction to Genome Data Analysis
3. Combining In-Memory Technology with Genome Data Analysis
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - IMDB Analysis Features for Applications
Pipeline Execution – Bringing the Model to Life

- Parser converts XML into executable, directed graph of task objects

- Execution environment is cluster of worker machines coordinated by scheduler

- Each task object has a corresponding implementation, e.g. tool invocation
Pipeline Execution – Worker Framework
## Pipeline Execution – Worker Framework

<table>
<thead>
<tr>
<th>ID</th>
<th>STATUS</th>
<th>PIPELINE ID</th>
<th>PARAMETERS</th>
<th>CREATED_AT</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.655</td>
<td>3</td>
<td>69</td>
<td>&quot;filename&quot;: &quot;SRR3945858.filt_fasta__2263&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 2263, &quot;priority&quot;: 0</td>
<td>07.08.2014 15:42:01.021</td>
<td>74</td>
</tr>
<tr>
<td>1.640</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;406.fastq__2263&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 2263, &quot;priority&quot;: 0</td>
<td>05.08.2014 13:47:31.818</td>
<td>3</td>
</tr>
<tr>
<td>1.639</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;BC_1.fasta__3228864&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 3228864, &quot;priority&quot;: 0</td>
<td>21.07.2014 13:01:06.877</td>
<td>75</td>
</tr>
<tr>
<td>1.638</td>
<td>2</td>
<td>73</td>
<td>&quot;filename&quot;: &quot;s_G1_L001_L1_001.fastq_1__25000&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 25000, &quot;priority&quot;: 0</td>
<td>21.07.2014 10:28:32.352</td>
<td>68</td>
</tr>
<tr>
<td>1.637</td>
<td>2</td>
<td>70</td>
<td>&quot;filename&quot;: &quot;H-10950_S9_L001_R1_001.fastq_794380&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 794380, &quot;priority&quot;: 0</td>
<td>18.07.2014 00:17:32.737</td>
<td>3</td>
</tr>
<tr>
<td>1.636</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;L21_500002_2.fastq&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 500002, &quot;priority&quot;: 0</td>
<td>20.06.2014 11:46:39.73</td>
<td>59</td>
</tr>
<tr>
<td>1.634</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;BC_1.fasta__3228864&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 3228864, &quot;priority&quot;: 0</td>
<td>12.06.2014 13:10:08.204</td>
<td>3</td>
</tr>
<tr>
<td>1.633</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;H-10950_S9_L001_R1_001.fastq_794380&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 794380, &quot;priority&quot;: 0</td>
<td>12.06.2014 13:09:25.584</td>
<td>3</td>
</tr>
<tr>
<td>1.632</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;H-10950_S9_L001_R1_001.fastq_794380&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 794380, &quot;priority&quot;: 0</td>
<td>12.06.2014 13:09:19.777</td>
<td>3</td>
</tr>
<tr>
<td>1.631</td>
<td>2</td>
<td>22</td>
<td>&quot;filename&quot;: &quot;H-10950_S9_L001_R1_001.fastq_794380&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 794380, &quot;priority&quot;: 0</td>
<td>12.06.2014 13:09:36.365</td>
<td>3</td>
</tr>
<tr>
<td>1.630</td>
<td>2</td>
<td>73</td>
<td>&quot;filename&quot;: &quot;SRR3945858.filt_fasta__2263&quot;, &quot;typ&quot;: &quot;file&quot;, &quot;read_count&quot;: 2263, &quot;priority&quot;: 0</td>
<td>10.06.2014 17:03:37.696</td>
<td>68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLUBTASK</th>
<th>TASK</th>
<th>STATUS</th>
<th>JOB</th>
<th>PARAMETER</th>
<th>UPDATED_AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>82.334</td>
<td>1.639</td>
<td>0</td>
<td>BamSort</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;0kr6909v00m0jvn.bam&quot;</td>
<td>2014-07-21 13:34:28.865</td>
</tr>
<tr>
<td>82.334</td>
<td>1.639</td>
<td>1</td>
<td>BamSort</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;0kr6909v00m0jvn.bam&quot;</td>
<td>2014-07-21 13:34:43.427</td>
</tr>
<tr>
<td>82.334</td>
<td>1.639</td>
<td>2</td>
<td>BamSort</td>
<td>&quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:15.317</td>
</tr>
<tr>
<td>82.335</td>
<td>1.639</td>
<td>0</td>
<td>BamIndex</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:15.333</td>
</tr>
<tr>
<td>82.335</td>
<td>1.639</td>
<td>1</td>
<td>BamIndex</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:15.836</td>
</tr>
<tr>
<td>82.335</td>
<td>1.639</td>
<td>2</td>
<td>BamIndex</td>
<td>&quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:17.551</td>
</tr>
<tr>
<td>82.336</td>
<td>1.639</td>
<td>0</td>
<td>Indel_Calling...</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:17.563</td>
</tr>
<tr>
<td>82.336</td>
<td>1.639</td>
<td>1</td>
<td>Indel_Calling...</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:34:17.892</td>
</tr>
<tr>
<td>82.336</td>
<td>1.639</td>
<td>2</td>
<td>Indel_Calling...</td>
<td>&quot;filename&quot;: &quot;hmgr3w4xchx5c.indel.vcf&quot;</td>
<td>2014-07-21 13:34:12.016</td>
</tr>
<tr>
<td>82.337</td>
<td>1.639</td>
<td>0</td>
<td>SNP_Calling...</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:32:12.033</td>
</tr>
<tr>
<td>82.337</td>
<td>1.639</td>
<td>1</td>
<td>SNP_Calling...</td>
<td>&quot;number_of_instances&quot;: 1, &quot;filename&quot;: &quot;cbts8lwvuy5ses.bam&quot;</td>
<td>2014-07-21 13:32:12.59</td>
</tr>
<tr>
<td>82.337</td>
<td>1.639</td>
<td>2</td>
<td>SNP_Calling...</td>
<td>&quot;filename&quot;: &quot;cdskoc0fsu4w760.snps.vcf&quot;</td>
<td>2014-07-21 13:32:19.045</td>
</tr>
</tbody>
</table>
Pipeline Execution –
Task Implementation

- Task implementation imported as modules to worker at runtime

- One super class for administrative things, all tasks implement particular method

```python
class BamSort(Task):
    def do_execute(self):
        input_filename = self.get_input("filename")

        result_filename = self.new_filename()
        self.system_command("samtools sort {0} {1}".format(input_filename, result_filename))

        self.add_output("filename", result_filename)
```
Pipeline Execution – Scheduler

- Scheduler is responsible for holding the structure of task objects
- Starts task when all predecessors are finished
- High availability of scheduler by storing global pipeline status in IMDB
  - In case of scheduler crash another worker can take scheduler role without any delay
- Scheduler uses workload information and execution statistics based on logs in IMDB
Different scheduling algorithms
- First-come first-served
- Lottery
- Shortest task first
- Priority-based assignment
- User-based assignment

Prioritize tasks to maximize utilization of workers
(Intermediate) results are imported into database
Pipelines –
Traditional vs. IMDB-supported Approach
Agenda

1. Introduction to In-Memory Technology
2. Introduction to Genome Data Analysis
3. Combining In-Memory Technology with Genome Data Analysis
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - IMDB Analysis Features for Applications
IMDB Technology for Genome Data Analysis – Alignment

- Extension of IMDB platform via own Genomics server
- Index creation at server start and storage in main memory
- Efficient processing via vectorization and bit parallelism
- Efficient streaming capabilities provided by IMDB platform

Multi-Core and Parallelization
Partitioning
Lightweight Compression
No Disk
Alignment – Performance

- First evaluations with Burrows-Wheeler-Alignment (BWA) as one representative of popular alignment algorithms
- Time saving up to a factor of 21 compared to BWA
- Alignment of low-coverage (20x) whole genome on cluster
  - Up to 346M reads
  - Alignment of all reads within ~1h
Variant Calling – Motivation

- Common variant calling tools all process files residing on disk space
  - Slow storage media
  - Large data files, e.g. >100GB per individual

- Idea: Access data from main memory and profit from built-in database features
  - Partitioning
  - Multi-core and parallelization
  - Lightweight compression

Chart 49
Variant Calling – Data

- Reference genome: Base sequence for comparison

- Read alignments: Reads from reconstructed sample genome

- All data is imported into database beforehand, with implicit
  - Data indexing
  - Lightweight compression

- Variant calling results conform to standard format and can easily be exported from database or used for further analyses

Chart 50
Variant Calling – Extending the Database Core

- Implementation as Application Function Library (AFL)
- Variant calling per chromosome
- Parallelization with MapReduce-like approach
- Invocation via stored procedure call

```sql
CALL "SYS_AFL"."VARCALL_AREA_CALL_SNP_VARIANTS_PROC"
    (SAMIMPORT.NA19240, GENES.HG19CHR22, 'chr22', 20, 20, 30, 40, VARIANTS.OUTPUTTAB) WITH OVERVIEW;
```
Variant Calling – Performance

- Built-in database functionalities simplify and speed up data preprocessing
- Average time saving of factor 22 compared to standard tools at equal accuracy
- SNP calling of high-coverage (64x) whole genome on cluster
  - 873M read alignments
  - ~18min
Agenda

1. Introduction to In-Memory Technology
2. Introduction to Genome Data Analysis

3. **Combining In-Memory Technology with Genome Data Analysis**
   - Pipeline Modeling
   - Pipeline Execution
   - IMDB Technology for Genome Data Analysis
   - **IMDB Analysis Features for Applications**
IMDB Analysis Features for Applications – Textual Analysis of Medical Documents

- IMDB provides text analysis features, e.g.
  - Fulltext indexing
  - Entity Recognition
  - Tokenization
  - Fuzzy search

- Mechanisms can be made domain-specific by specifying
  - Dictionaries
  - CGUL rules containing regular expressions with linguistic attributes
1. Specify dictionary in XML and/or CGUL rules:

```xml
<?xml version='1.0' encoding='UTF-8'>
<dictionary xmlns="http://www.sap.com/ta/4.0">
  <entity_category name="BODY_PART_ORGAN_OR_ORGAN_COMPONENT">
    <entity_name name="Abdominal wall muscle" type="PF"/>
    <entity_name name="Muscle of abdomen" type="PF"/>
    <entity_name name="Skeletal muscle structure of abdomen" type="PF"/>
    <entity_name name="Abdominal wall muscle" type="VO"/>
    <entity_name name="Muscle of abdomen" type="VO"/>
    <entity_generation language="english" type="standard"/>
  </entity_category>
  [...]
</dictionary>
```

2. Compile XML dictionary for database and reference them in config file
3. Create fulltext index:

```sql
CREATE FULLTEXT INDEX "EXAMPLE"."EXAMPLE_INDEX" ON EXAMPLE"."EXAMPLE_DATA" ("TEXT")
CONFIGURATION 'PROJECT::MED_TERMS' ASYNC LANGUAGE DETECTION ('EN')
FUZZY SEARCH INDEX ON TEXT ANALYSIS ON TOKEN SEPARATORS '\'/;,.:-_()[]<>!?*@+{}="&'
```

4. Get results in database table and use for further analyses:

<table>
<thead>
<tr>
<th>TA_RULE</th>
<th>TA_TOKEN</th>
<th>TA_LANGUAGE</th>
<th>TA_NORMALIZED</th>
<th>TA_PARAGRAPH</th>
<th>TA_SENTENCE</th>
<th>TA_CREATED_AT</th>
<th>TA_OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Extraction</td>
<td>woman</td>
<td>en</td>
<td>patient</td>
<td>3</td>
<td>3</td>
<td>11.06.2014 16:38:28.779</td>
<td>473</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>man</td>
<td>en</td>
<td>patient</td>
<td>3</td>
<td>3</td>
<td>11.06.2014 16:38:28.827</td>
<td>437</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>Inpatient</td>
<td>en</td>
<td>patient</td>
<td>1</td>
<td>3</td>
<td>11.06.2014 16:38:28.827</td>
<td>211</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>woman</td>
<td>en</td>
<td>patient</td>
<td>3</td>
<td>3</td>
<td>11.06.2014 16:38:28.827</td>
<td>460</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>Inpatient</td>
<td>en</td>
<td>patient</td>
<td>2</td>
<td>3</td>
<td>11.06.2014 16:38:28.827</td>
<td>223</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>Inpatient</td>
<td>en</td>
<td>patient</td>
<td>1</td>
<td>1</td>
<td>11.06.2014 16:38:28.827</td>
<td>227</td>
</tr>
<tr>
<td>Entity Extraction</td>
<td>Patient</td>
<td>en</td>
<td>patient</td>
<td>4</td>
<td>4</td>
<td>11.06.2014 16:38:28.865</td>
<td>273</td>
</tr>
</tbody>
</table>
IMDB Analysis Features for Applications – Statistical Analyses Functions

- IMDB provides specific **analysis functions** tightly integrated within the database, e.g. k-means or hierarchical clustering

- Highly parallelized and efficient using database framework

- Invoked as stored procedures via SQL statement:

  ```sql
  CALL _SYS_AFL.PAL_HC(DATA_TAB, PARAM_TAB, COMBINEPROCESS_TAB, RESULT_TAB);
  ```
Analyze Genomes – An In-Memory Computing Platform

Cohort Analysis  Pathway Topology Analysis  Publication Search  Clinical Trial Assessment  Genome Browser  ...

Analytical Tools  Access Control  App Store  Statistical Tools  ...

Billing  Extensions  ...

Genome Data  Publications  Genome Metadata  Pipeline Models  ...

Pathways  Data  In-Memory Database

Keynote Wednesday morning!
Keep in contact with us.

Cindy Fähnrich, M. Sc.  
cindy.faehnrich@hpi.de

Dr. Matthieu-P. Schapranow  
schapranow@hpi.de  
http://we.analyzegenomes.com/

Hasso Plattner Institute  
Enterprise Platform & Integration Concepts  
August-Bebel-Str. 88  
14482 Potsdam, Germany
Medical Knowledge Cockpit for Clinicians
Pathway Topology Analysis

- Search in pathways is limited to “is a certain element contained” today
- Integrated >1,5k pathways from international sources, e.g. KEGG, HumanCyc, and WikiPathways, into HANA
- Implemented graph-based topology exploration and ranking based on patient specifics
- Enables interactive identification of possible dysfunctions affecting the course of a therapy before its start

**Unified access** to multiple formerly disjoint data sources

**Pathway analysis** of genetic variants with graph engine