



Stratosphere

Parallel Analytics in the Cloud beyond Map/Reduce

14th International Workshop
on High Performance Transaction Systems (HPTS)


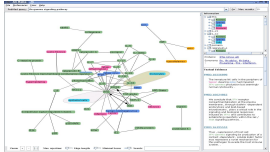
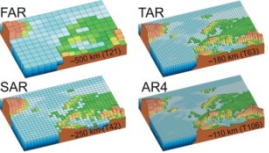
Poster Sessions, Mon Oct 24 2011

Thomas Bodner
T.U. Berlin

The Stratosphere Project*



Use-Cases




Scientific Data Life Sciences Linked Data



StratoSphere Query Processor
Above the Clouds

Infrastructure as a Service



- Explore the power of Cloud computing for complex information management applications
- Database-inspired approach
- Analyze, aggregate, and query
- Textual and (semi-) structured data
- Research and prototype a web-scale data analytics infrastructure

* FOR 1306: DFG funded collaborative project among TU Berlin, HU Berlin and HPI Potsdam

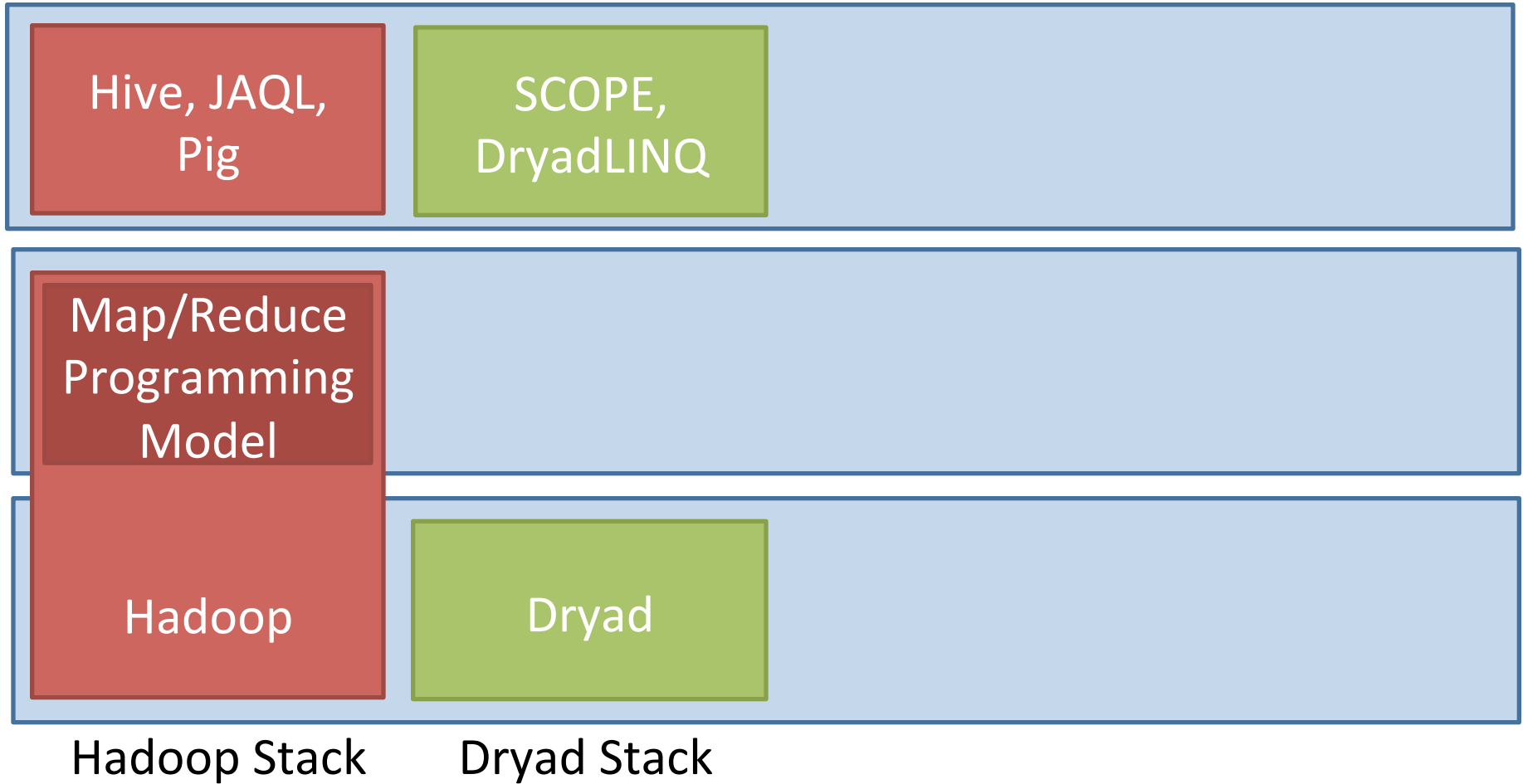


- Architecture of the Stratosphere System
- The PACT Programming Model
- The Nephele Execution Engine
- Demonstration of an Example Job

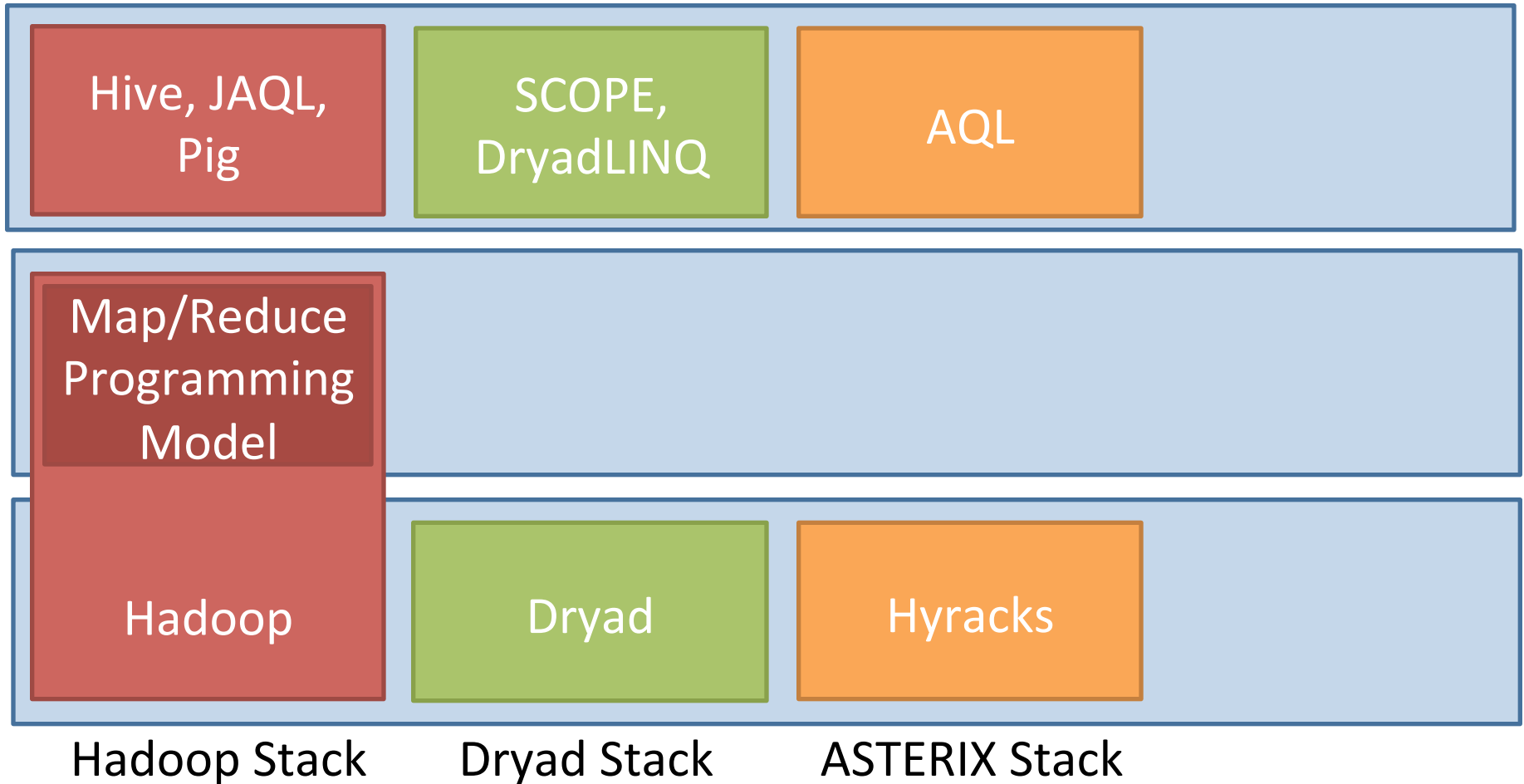


Hadoop Stack

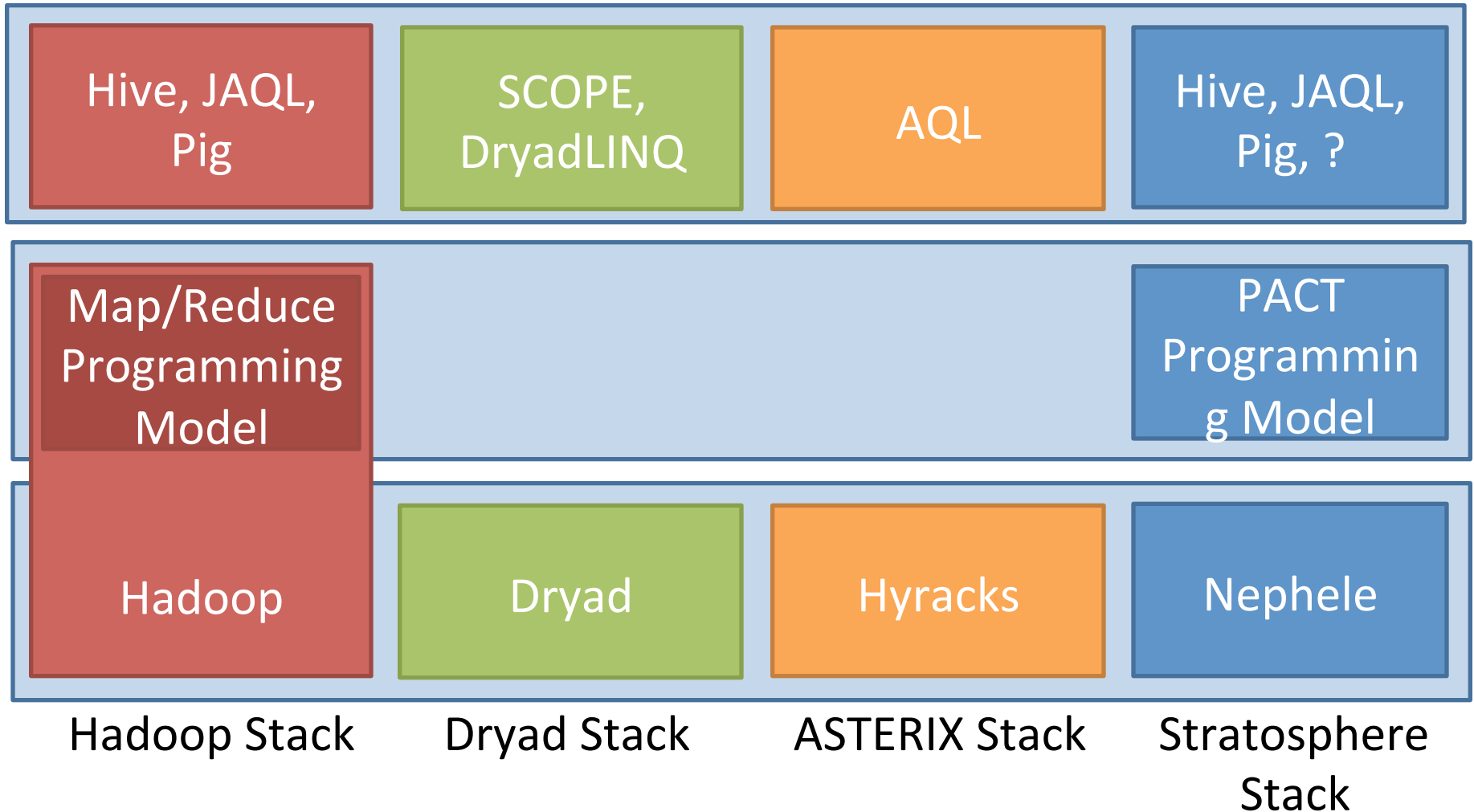
Architecture Overview



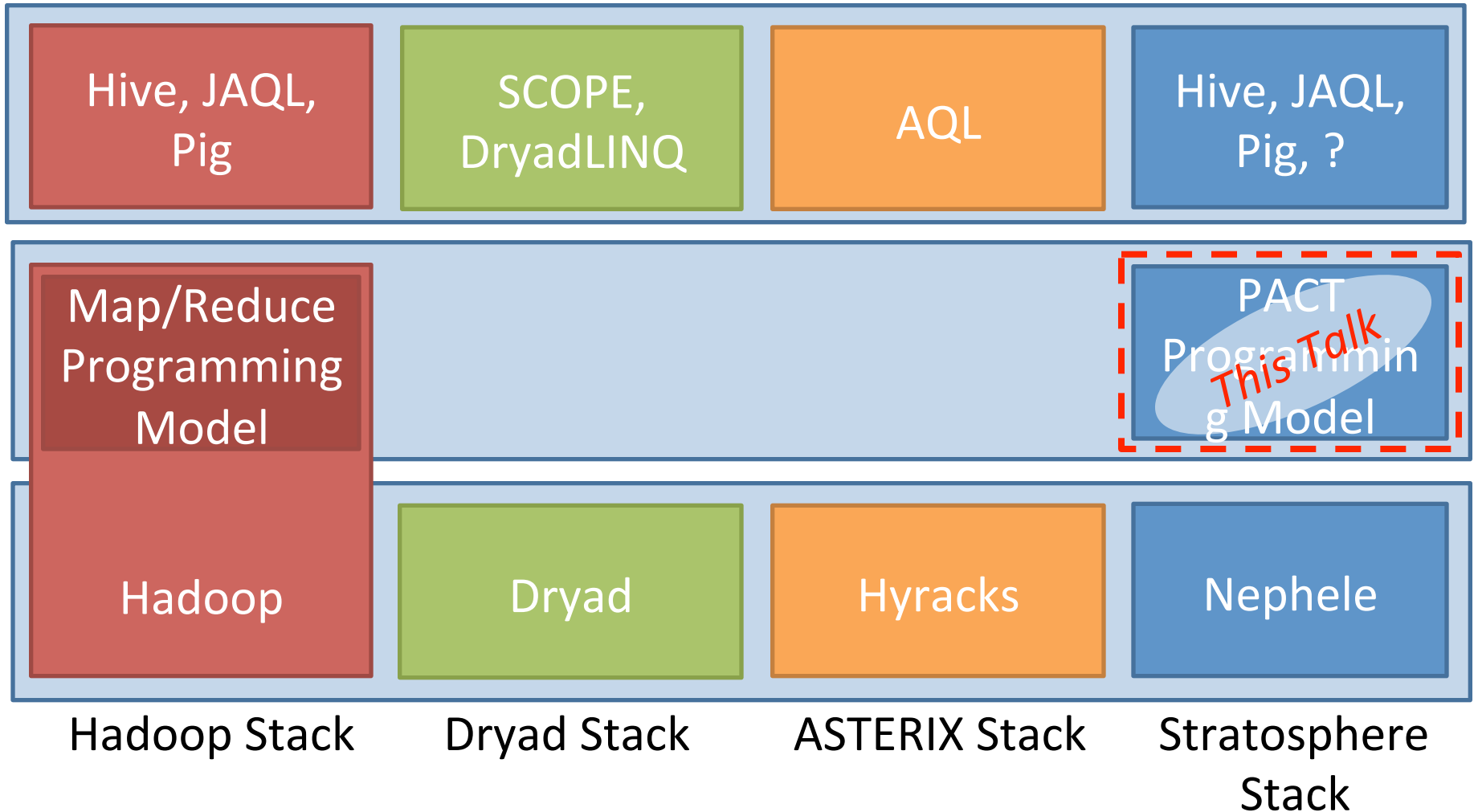
Architecture Overview



Architecture Overview

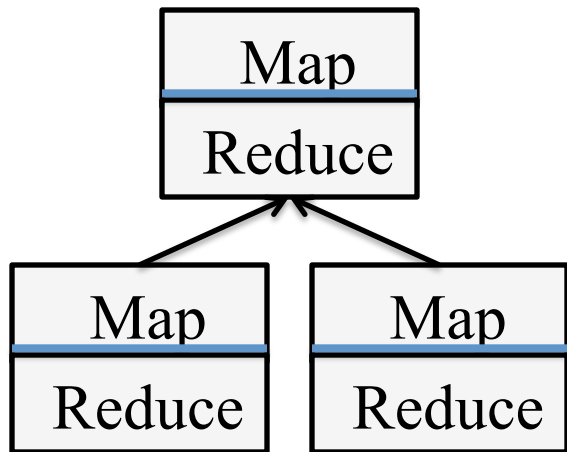


Architecture Overview



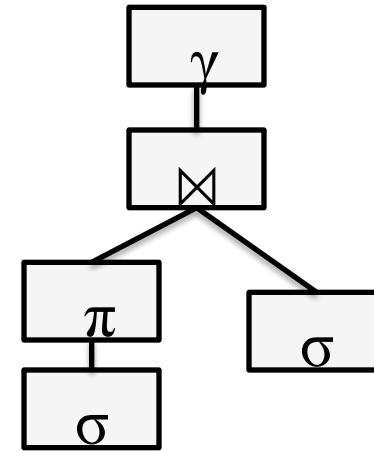


Map / Reduce



- Schema Free
- Many semantics hidden inside the user code (tricks required to push operations into map/reduce)
- Single default way of parallelization

Relational Databases



- Schema bound (relational model)
- Well defined properties and requirements for parallelization
- Flexible and optimizable

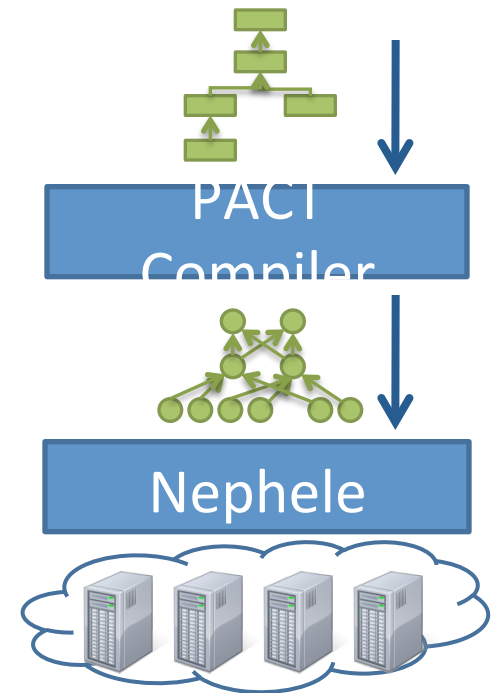


GOAL: Advance the M/R Programming Model

Stratosphere in a Nutshell



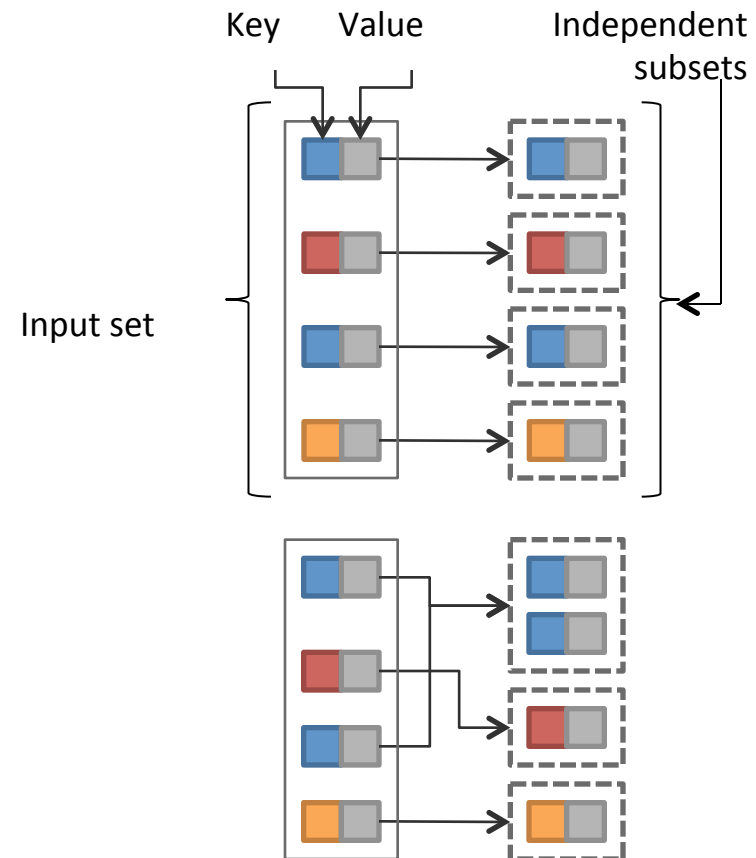
- PACT Programming Model
 - Parallelization Contract (PACT)
 - Declarative definition of data parallelism
 - Centered around second-order functions
 - Generalization of map/reduce
- Nephele
 - Dryad-style execution engine
 - Evaluates dataflow graphs in parallel
 - Data is read from distributed filesystem
 - Flexible engine for complex jobs
- Stratosphere = Nephele + PACT
 - Compiles PACT programs to Nephele dataflow graphs
 - Combines parallelization abstraction and flexible execution
 - Choice of execution strategies gives optimization potential



Intuition for Parallelization Contracts



- Map and reduce are second-order functions
 - Call first-order functions (user code)
 - Provide first-order functions with subsets of the input data
- Define dependencies between the records that must be obeyed when splitting them into subsets
 - Required partition properties
- Map
 - All records are independently processable
- Reduce
 - Records with identical key must be processed together

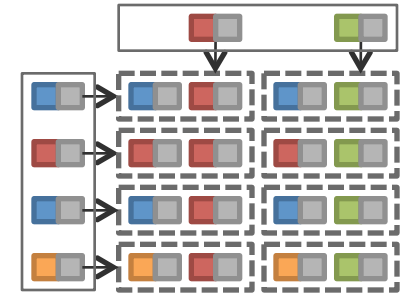


Contracts beyond Map and Reduce



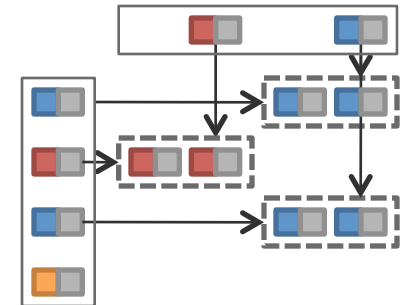
■ Cross

- Two inputs
- Each combination of records from the two inputs is built and is independently processable



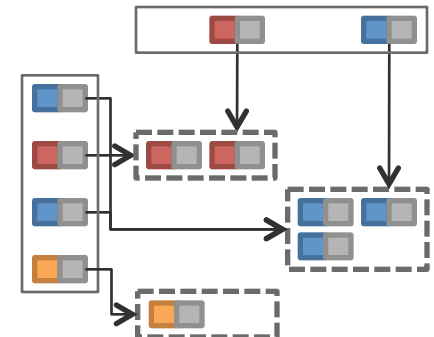
■ Match

- Two inputs, each combination of records with equal key from the two inputs is built
- Each pair is independently processable



■ CoGroup

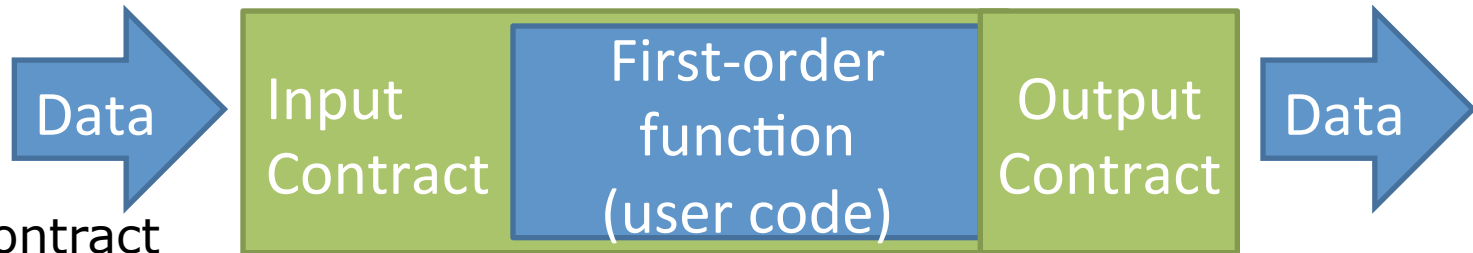
- Multiple inputs
- Pairs with identical key are grouped for each input
- Groups of all inputs with identical key are processed together



Parallelization Contracts (PACTs)



- Second-order function that defines properties on the input and output data of its associated first-order function



- Input Contract

- Specifies dependencies between records (a.k.a. "What must be processed together?")
- Generalization of map/reduce
- Logically: Abstracts a (set of) communication pattern(s)
 - For "reduce": repartition-by-key
 - For "match" : broadcast-one or repartition-by-key

- Output Contract

- Generic properties preserved or produced by the user code
 - key property, sort order, partitioning, etc.
- Relevant to parallelization of succeeding functions

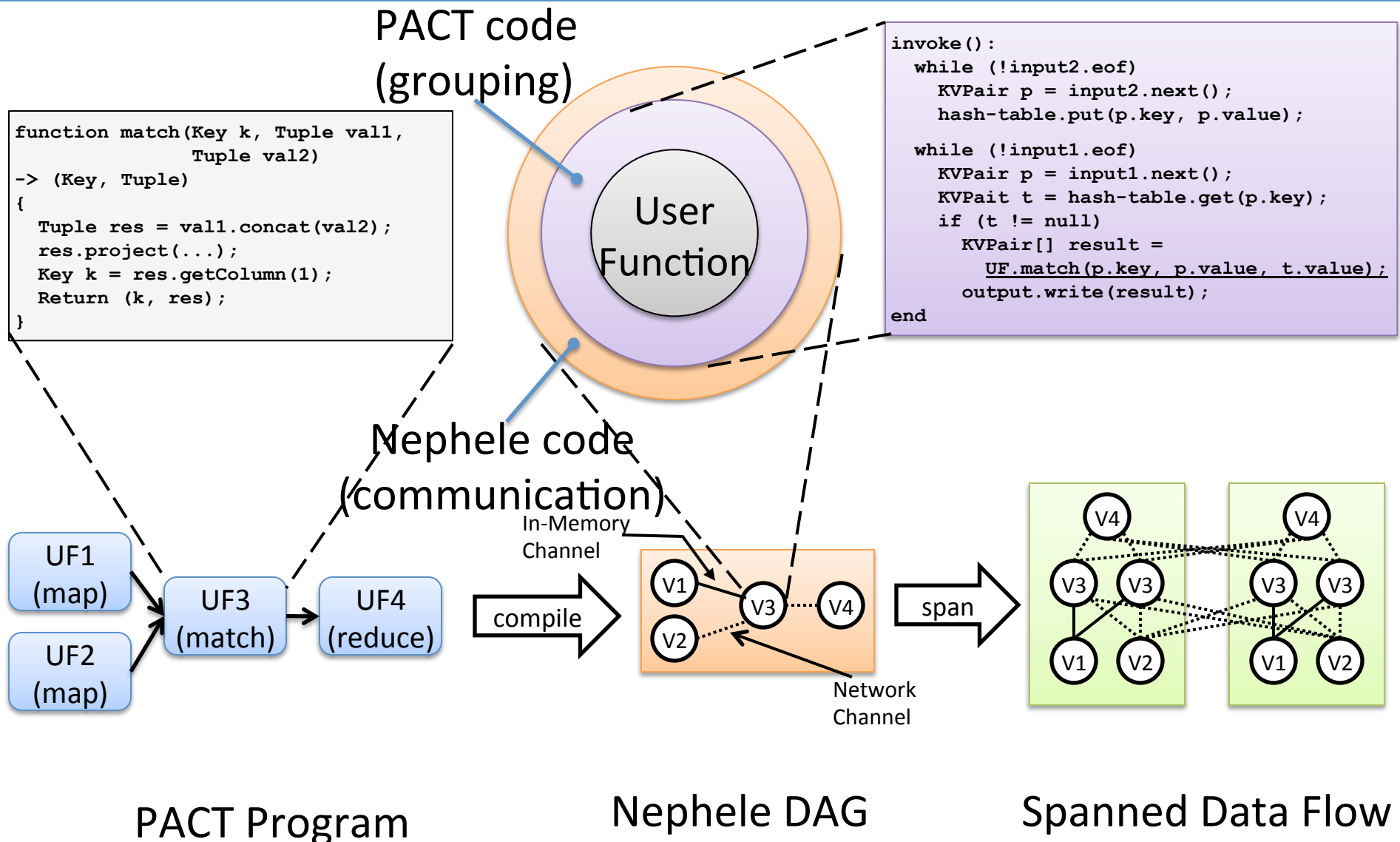


- For certain PACTs, several distribution patterns exist that fulfill the contract
 - Choice of best one is up to the system

- Created properties (like a partitioning) may be reused for later operators
 - Need a way to find out whether they still hold after the user code
 - Output contracts are a simple way to specify that
 - Example output contracts: Same-Key, Super-Key, Unique-Key

- Using these properties, optimization across multiple PACTs is possible
 - Simple System-R style optimizer approach possible

From PACT Programs to Data Flows





- www.stratosphere.eu provides publications, open-source release and examples
- *„Nephele: Efficient Parallel Data Processing in the Cloud“*, D. Warneke et al., MTAGS 2009
- *„Nephele/PACTs: a programming model and execution framework for web-scale analytical processing“*, D. Battré et al., SoCC 2010
- *„Massively Parallel Data Analysis with PACTs on Nephele“*, A. Alexandrov et al. PVLDB 2010
- *„MapReduce and PACT - Comparing Data Parallel Programming Models“*, A. Alexandrov et al., BTW 2011