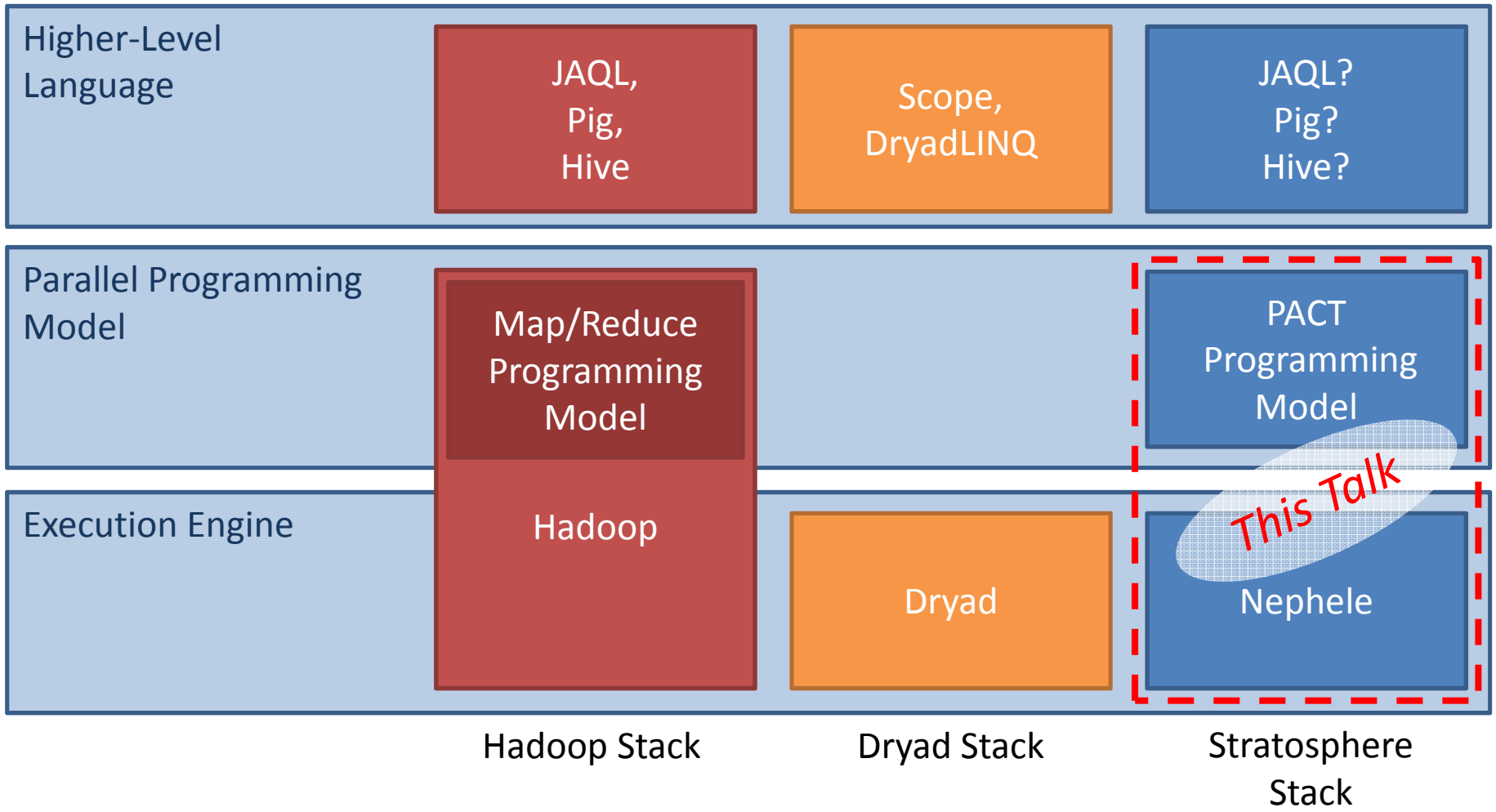
Cloud Computing
Tutorial Session 4

Björn Lohrmann
Complex and Distributed IT-Systems
Bjoern.lohrmann@tu-berlin.de

Agenda

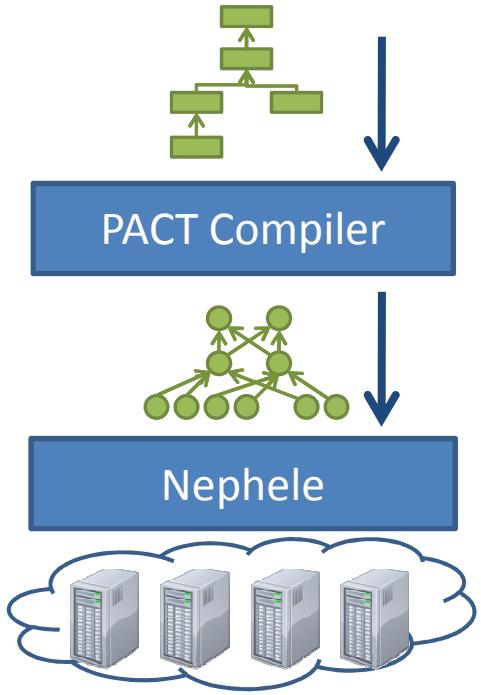
- Architecture of the Stratosphere System
  - The PACT Programming Model
  - The Nephele Execution Engine
- Project Assignment #4
- TPC-H benchmark data

Architecture Overview



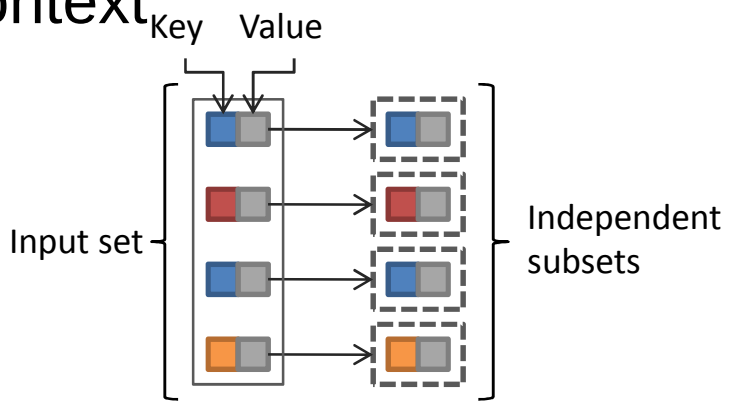
Stratosphere in a Nutshell

- 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

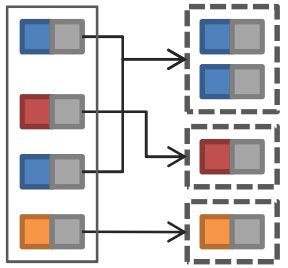


An Intuition for Parallelization Contracts (PACTs)

- Map and reduce are second-order functions
  - Call first-order functions (user code)
  - Provide first-order functions with subsets of the input data
- Map and reduce are PACTs in our context
- Map
  - All pairs are independently processed



- Reduce
  - Pairs with identical key are grouped
  - Groups are independently processed



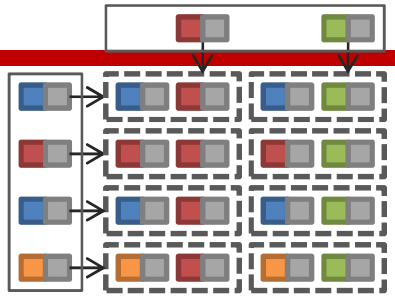
What is a PACT?

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

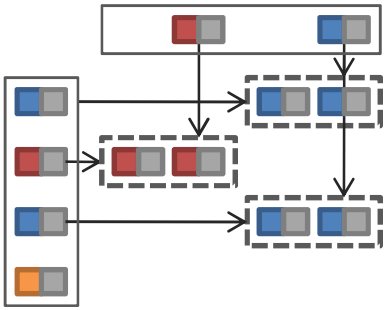
- Input Contract
  - Generates independently processable subsets of data
  - Generalization of map/reduce
  - Enforced by the system
- Output Contract
  - Describes properties of the output of the first-order function
  - Use is optional but enables certain optimizations
  - Guaranteed by the user
- Key-Value data model

PACTs beyond Map and Reduce

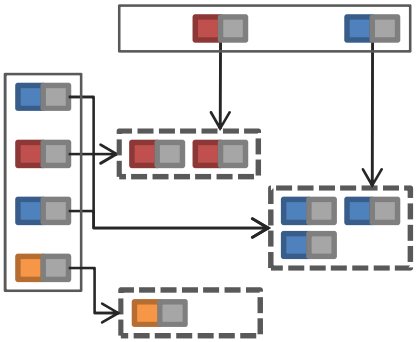
- Cross
  - Multiple inputs
  - Cartesian Product of inputs is built
  - All combinations are processed independently



- Match
  - Multiple inputs
  - All combinations of pairs with identical key over all inputs are built
  - All combinations are processed independently
  - Contract resembles an equi-join on the key



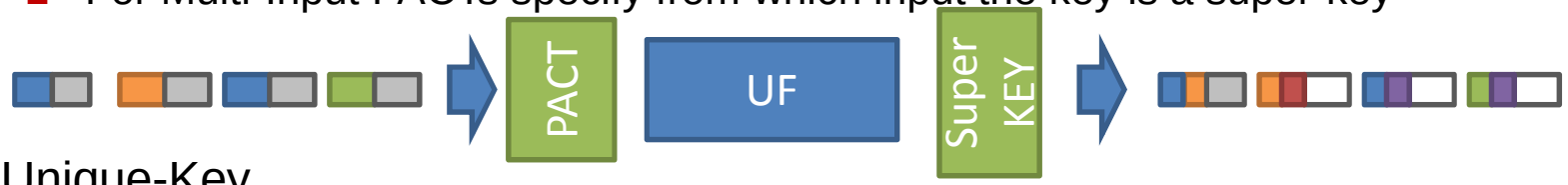
- CoGroup
  - Multiple inputs
  - Pairs with identical key are grouped for each input
  - Groups of all inputs with identical key are processed together



Output Contracts *(examples)*

- Same-Key
  - User Function does not alter the key
  - For Multi-Input PACTs specify whose input-key remains

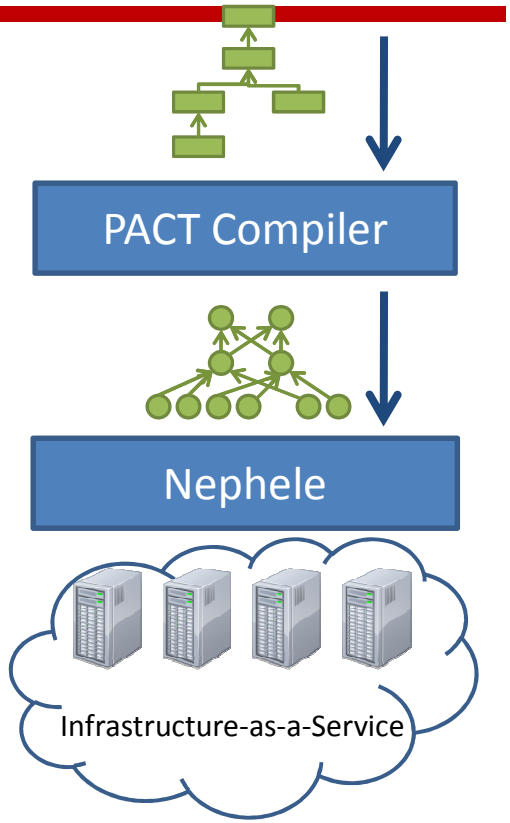
- Super-Key
  - Key generated by UF is a super-key of the input key
  - For Multi-Input PACTs specify from which input the key is a super-key



- Unique-Key
  - UF produces unique keys

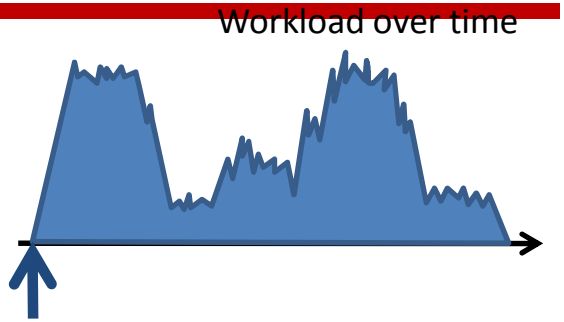
Nephele Execution Engine

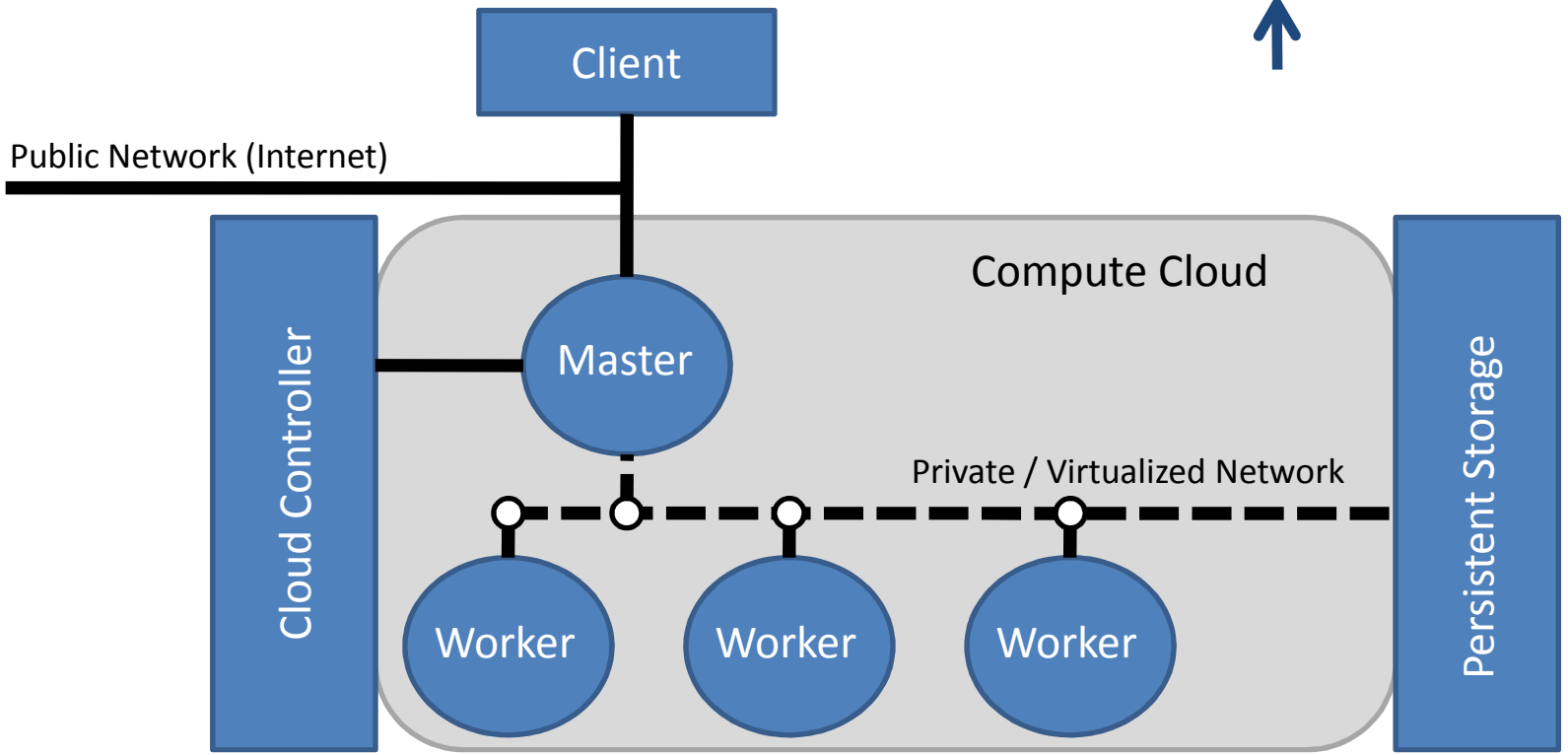
- Executes Nephele schedules
  - compiled from PACT programs
- Design goals
  - Exploit scalability/flexibility of clouds
  - Provide predictable performance
  - Efficient execution on 1000+ nodes
  - Introduce flexible fault tolerance mechanisms
- Inherently designed to run on top of an IaaS Cloud
  - Can exploit on-demand resource allocation
  - Heterogeneity through different types of VMs possible
  - Knows Cloud's pricing model



Nephele Architecture

- Standard master worker pattern
- Workers can be allocated on demand





Project Assignment #4

- Goal
  - Compare Hadoop and PACT programming models for a simple query
  - Use Hadoop and Stratosphere frameworks
- Test data
  - TPC-H benchmark tools
  - Generates relational data model in ASCII format

TPC-H Data Model

- Relations relevant for assignment
  - Customers
    - Have key, name and other attributes
  - Orders
    - Customers create orders
    - Has key, totalprice and other attributes
    - Has foreign key on customer
  - Lineitems
    - Orders consist of lineitems
    - Has foreign key on order