BioDynaMo Cloud Architecture

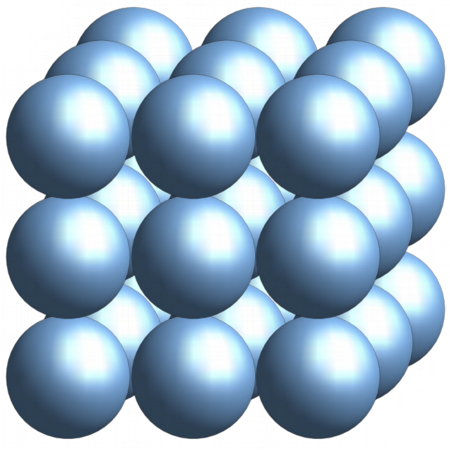
Lukas Breitwieser

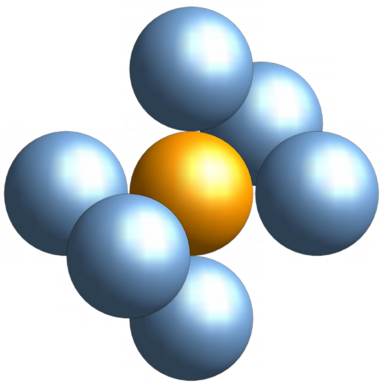
Outline

- Current Status
- Our Goal (in numbers)
- High Level Architecture
- Components in more detail
- State
- Failure scenarios
- Requirement for BLOB store
- Volunteer Computing

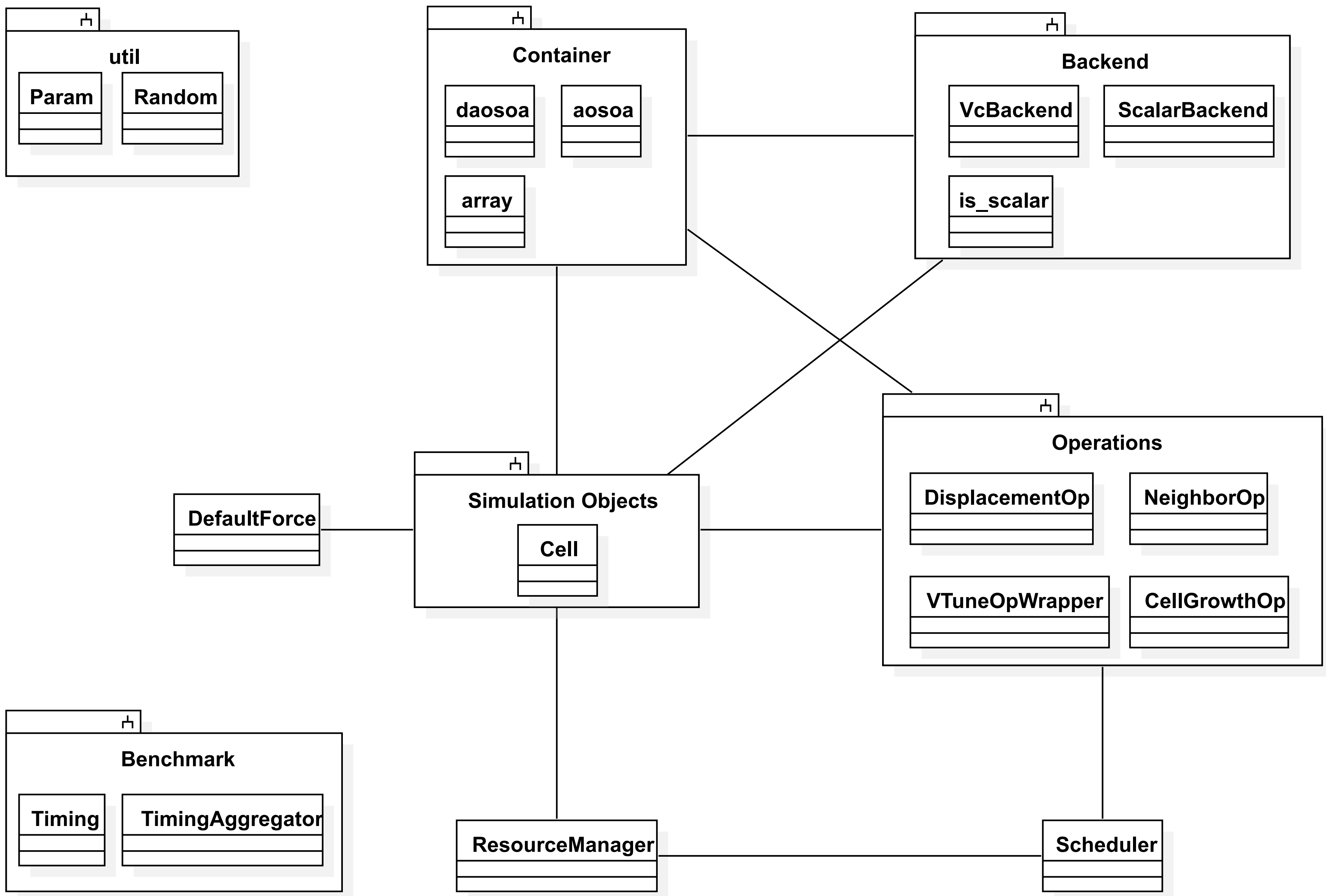
Parallel Prototype

- API completely different from ported BioDynaMo
- Only cell bodies, no neurites
- Only mechanical interaction

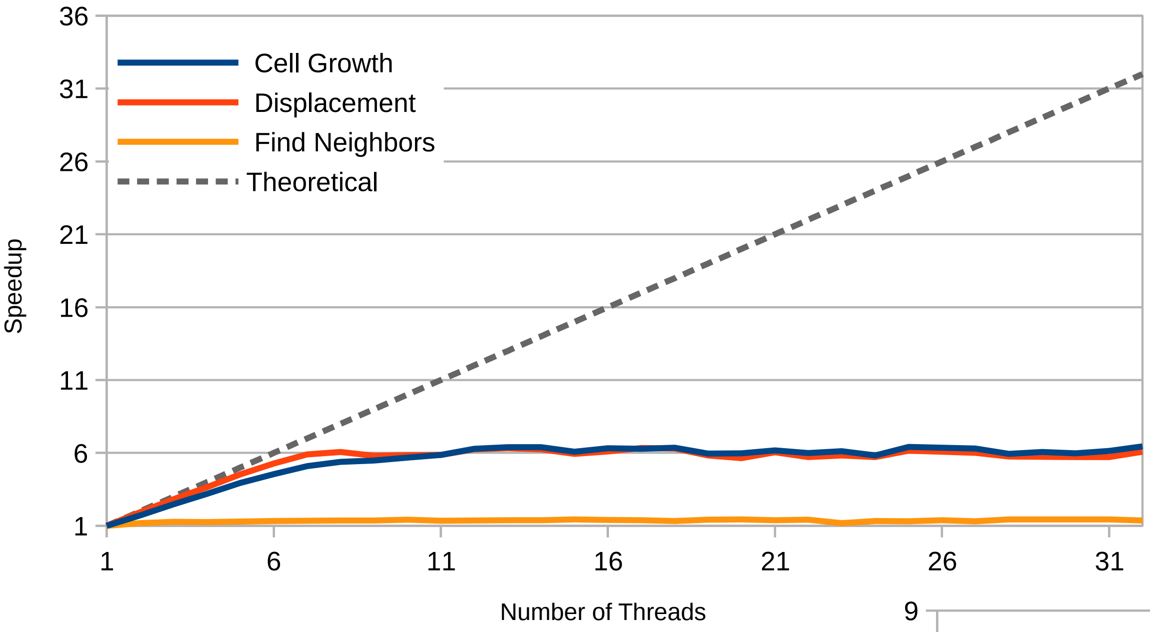


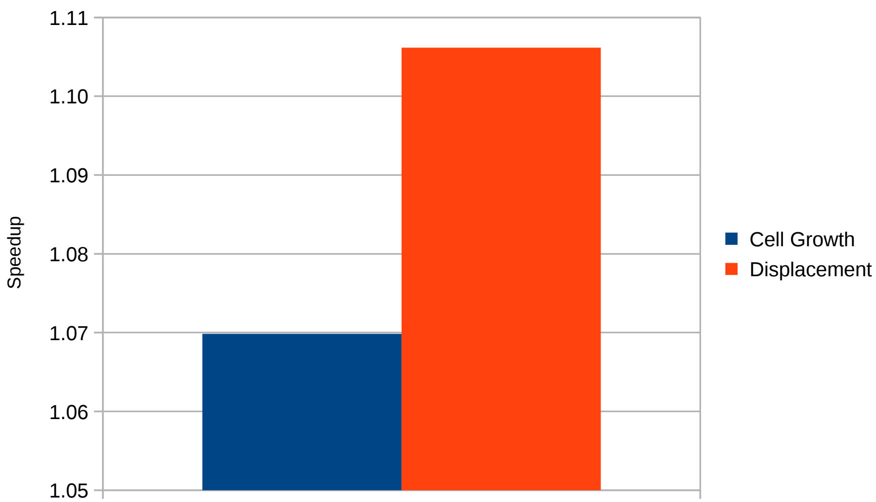


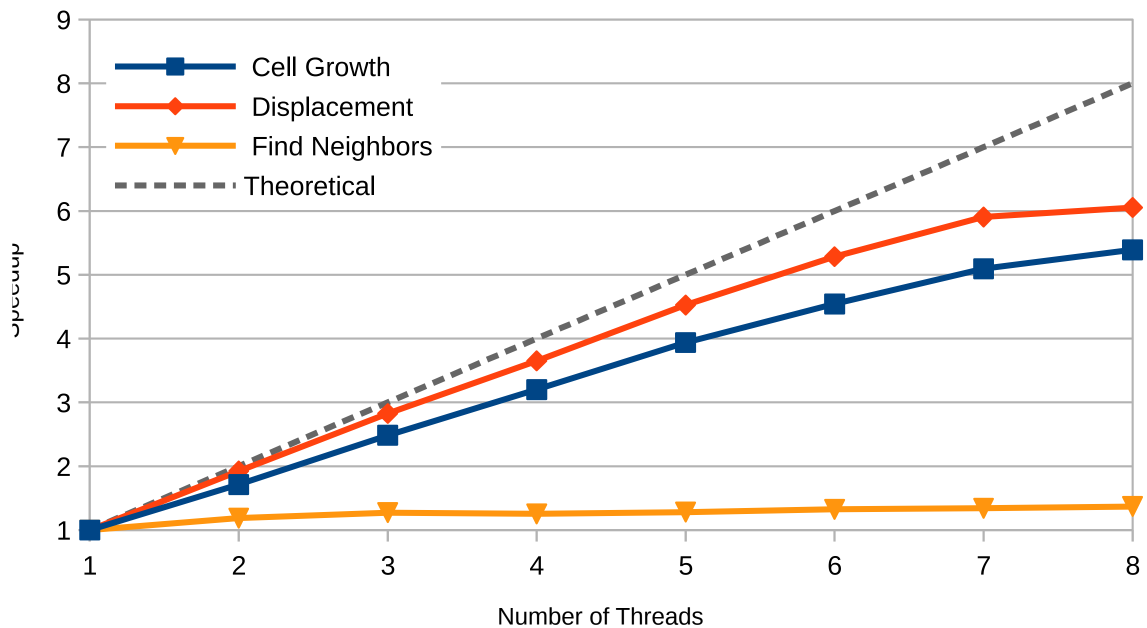
Parallel Prototype



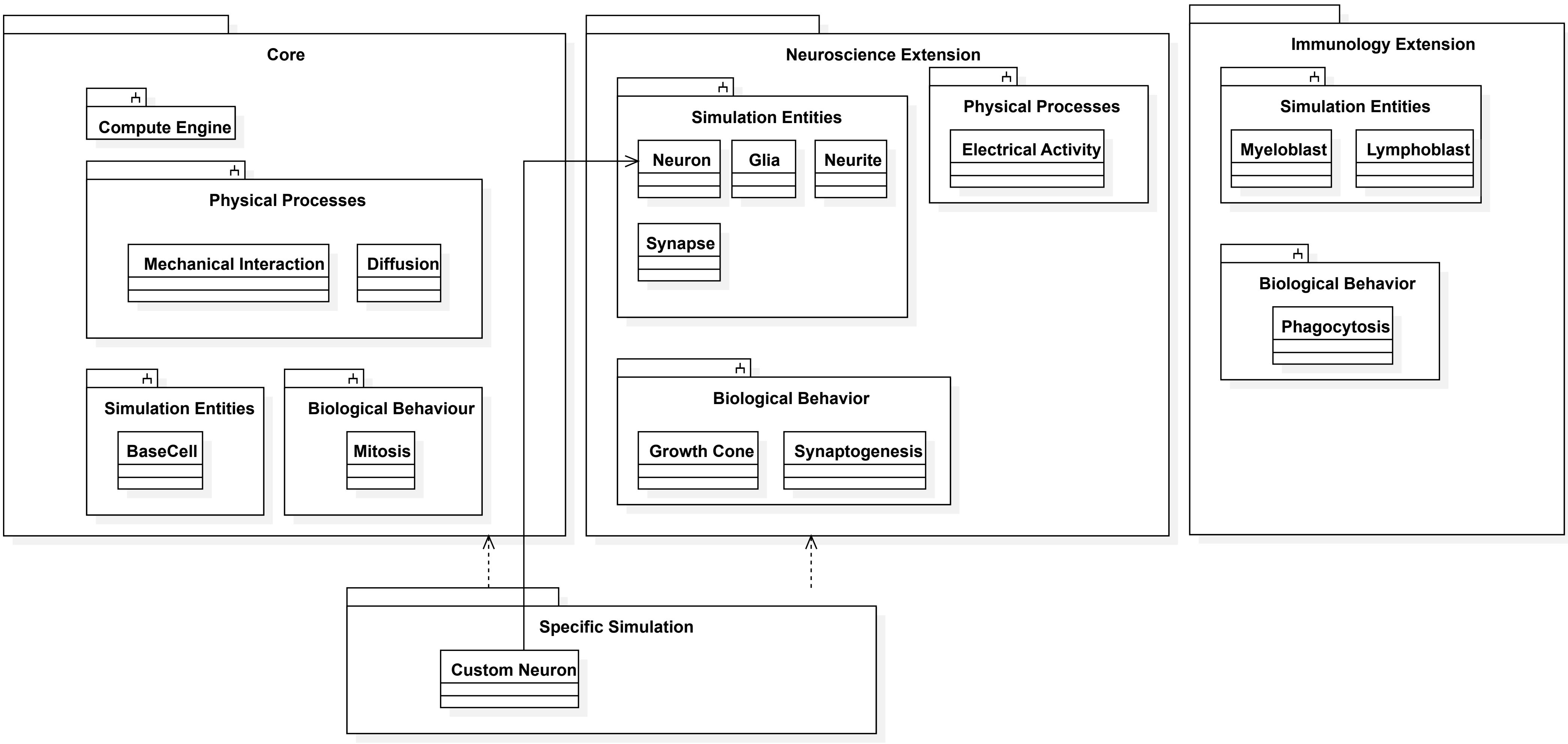
Parallel Prototype







Parallel Prototype



Other

- Flexibility concept exists
- Ahmad integrated IO into the **serial** BDM version

Our Goal

Run an agent based simulation in a distributed environment

Agents only see local environment

Difference to HPC:

- failures are the norm not the exception
- massive scale
- commodity hardware

Failures

server uptime without failure	3	years
	1095	days
Failures per hour is binomial r.v.		
Expectation E[X] = np	#machines	failures per day
	100	0.1
	1000	0.9
	10000	9.1
	100000	91.3

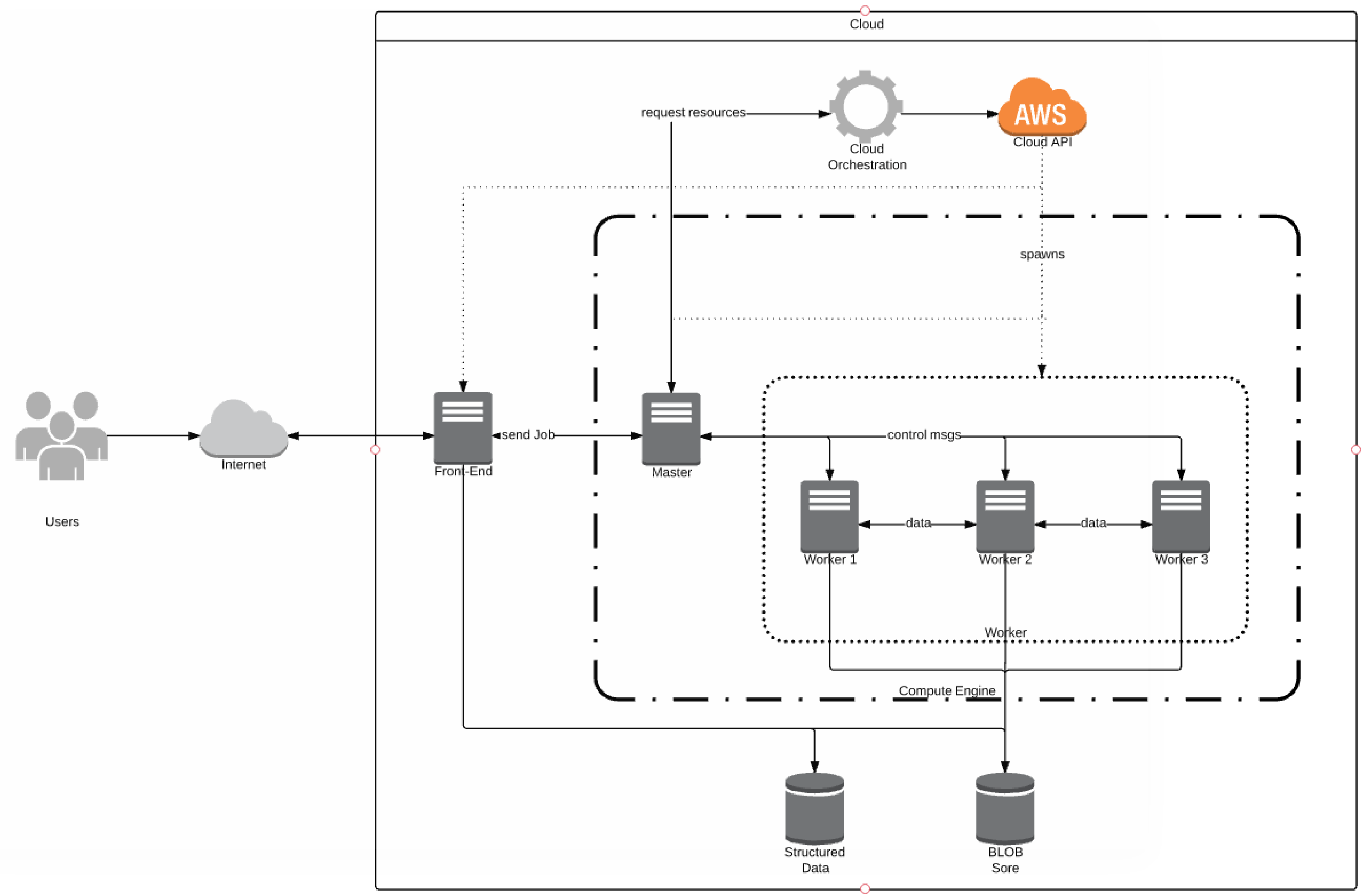
Computational Complexity

- Details: see spreadsheet
- Summary:
- Simulating 10% of the cortex requires:
  - 608 TB of memory
  - 10000 machines with 61 GB
  - Costs for one month 24x7 (AWS):
    - On demand: 5M $
    - Reserved: 2M $
  - Ghost area: 500 MB

transmission time [s]	Intra DC [10 Gbit]	Inter DC [100 Mbit]	Consumer [10 Mbit]
sec	0.40	39.87	397.89
min		0.66	6.63

BIODYNAMO CLOUD ARCHITECTURE

Lukas Brei



Top Level Components 1/2

- **Front End**
  - User defines simulation logic, initial simulation state and termination criterion and submits job (batch session);
  - Interactive session: user influences simulation state (modifies certain properties), create -, revert to snapshot, display intermediary results, ...
  - Front end gathers information and compiles it into Job which is sent to the Compute Engine (Master Node).
- **Simulation Job**
  contains simulation logic (code), initial simulation state and termination criterion; created by Front End; sent to Compute Engine
- **Master Node**
  - Requests resources and sends them to cloud orchestration (e.g. 100 nodes with >50GB RAM and 10GB Ethernet)
  - Membership List and Failure Detection
  - Scheduling (Partition of Space or assignment of tasks)
  - Synchronization coordination
  - Control Messages (request checkpoint, revert to snapshot, load balancing)

Top Level Components 2/2

- **Orchestration**

  receives requests from Master node and translates it into API calls for different cloud providers
- **Worker Node**
  - receives a certain volume it is responsible for
  - receives data from neighbor nodes
  - calculates on time step
  - sends ghost regions to neighbors
  - alternative task based:
    - receive set of tasks and input data
    - calculate tasks
    - send results to node with consecutive task
- **BLOB Store**

  stores large objects: checkpoints, results (final simulation state, videos, pictures)
- **Structured Data**

  User Data, Simulation Metadata

State

- Front – End:
  - HTTP session
  - unsaved inputs
- Structured Data
  - User Data
  - Simulation metadata
- BLOB Store
  - Simulation Results (final simulation state, pictures, videos, aggregated data)
  - Checkpoints
- Master Node:
  - Running Jobs
  - Volume partitioning
  - List of worker nodes
- Worker Nodes
  - Simulation Data

Failure Scenarios 1/3

- Persistent Storage (BLOB, Structured):
  - Event: disk failure
  - Result: All Data Lost → severity 10/10
  - Action: redundancy / fault tolerant storage
- Front End
  - Event: node fails
  - Result: Session Data and unsaved inputs → severity 1/10
  - Action: spawn new front end – (user has to log in again – some inputs might be lost)

Failure Scenarios 2/3

- Master Node
  - Event: process crashes
  - Result: Job data, running simulation data and worker membership data lost
  - Option 1: data lost: spawn new node, rediscover members, recalculate volume partition, assign new partition to workers (→ probably not the same as before → massive communication)
  - Option 2: data was persisted “somewhere”: spawn new node, read back data
- Workers
  - Event: e.g. process crashes
  - Result: simulation state of subvolume lost since last checkpoint
  - Option 1: global checkpoint
all worker nodes revert to last checkpoint – massive communication
  - Option 2: global checkpoint + neighbors store sequence of halo region
replace failed process; recalculate simulation since checkpoint only for subvolume – other workers: idle, other work
  - Option 3: have multiple workers calculate the same subvolume

Failure Scenarios 3/3

Network Failures

- Worker – Worker
  - worker retransmit with backoff; after some timeout; inform master
  - Result: long latency; worst case same as worker failure

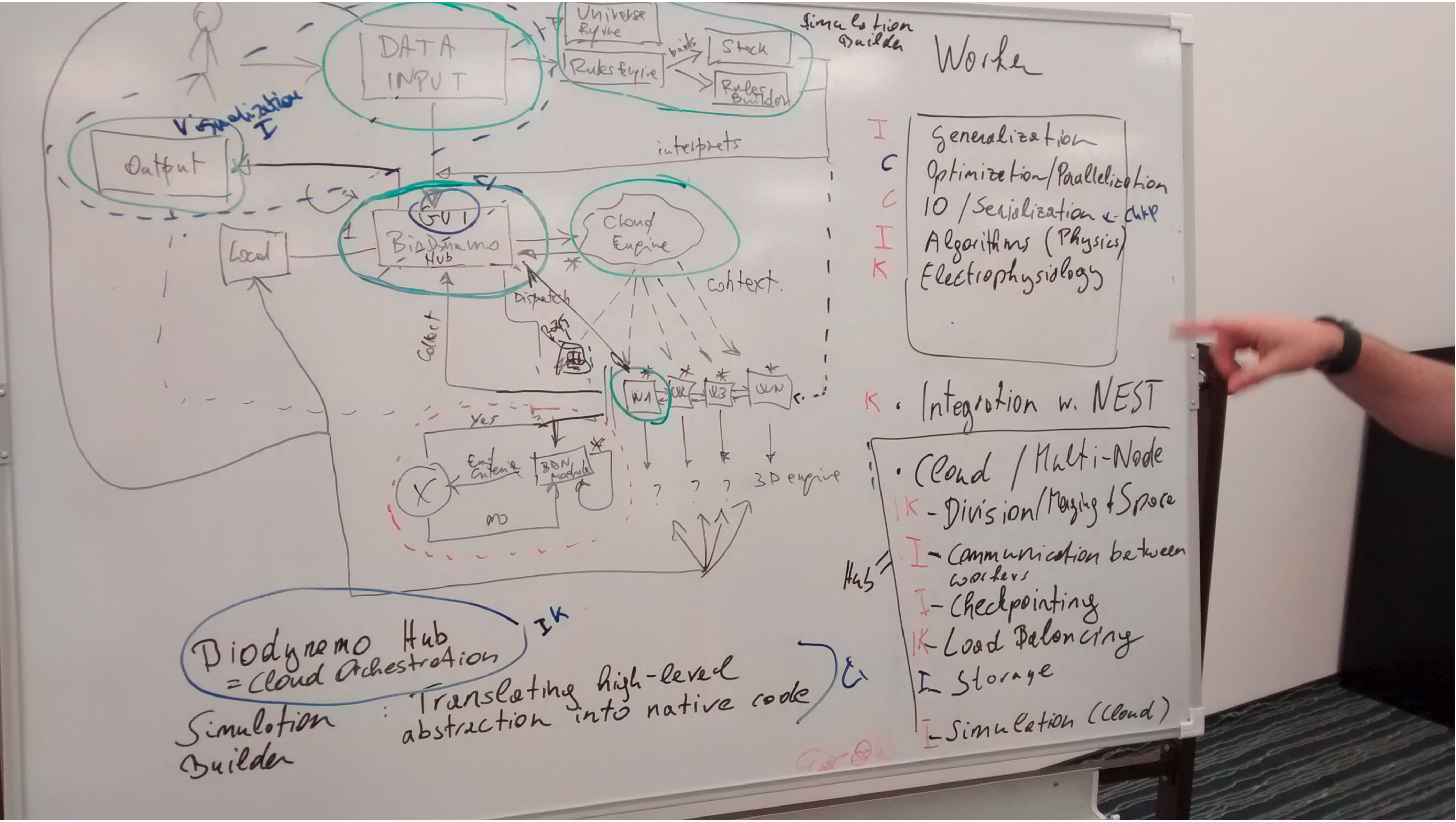
Requirements

BLOB Store:

- fault tolerant
- store 10s TB – PB
- data access pattern
  - during simulation: write/read checkpoint – write video / image
    → always read write whole file
    → HDFS or equivalent (HBase not needed)
  - after simulation (data / analysis): maybe retrieve subset (< 64MB)?

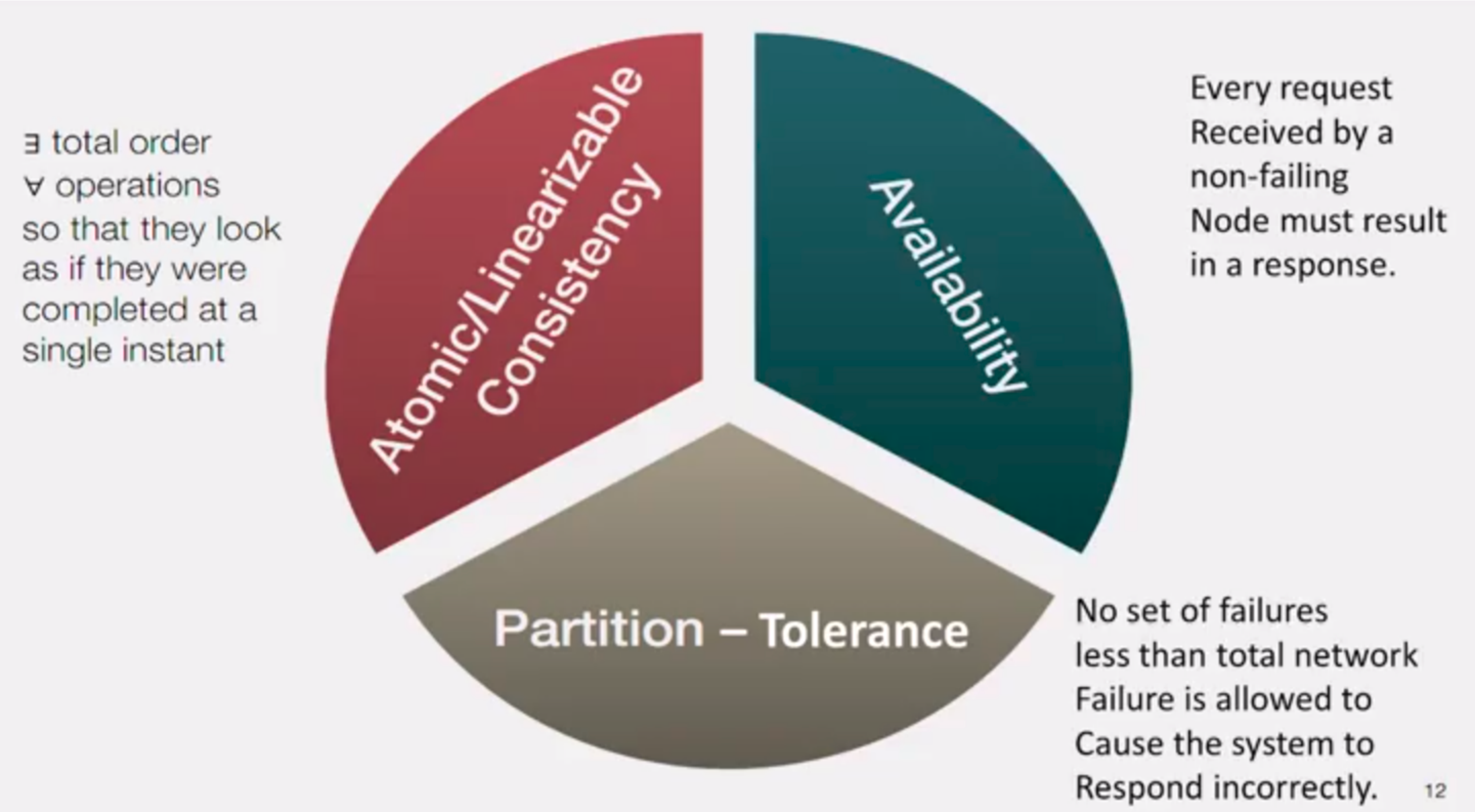
Backup Slides

Innopolis



DATA INPUT
Universe Engine
Rules Engine
Stack
Rules Builder
Simulation Builder
Worker
Visualization
Output
interprets
GUI
BioDynamo Hub
Cloud Engine
Local
context
Dispatch
Collect
W1
W3
WN
Yes
BDN module
no
3D engine
Generalization
Optimization/Parallelization
IO / Serialization
Algorithms (Physics)
Electrophysiology
Integration w. NEST
Cloud / Multi-Node
- Division/Merging of Space
- Communication between workers
- Checkpointing
- Load Balancing
- Storage
- Simulation (Cloud)
Hub
Biodynamo Hub = Cloud Orchestration
Simulation Builder : Translating high-level abstraction into native code

CAP-Theorem


∃ total order
∀ operations
so that they look
as if they were
completed at a
single instant
Atomic/Linearizable
Consistency
Availability
Every request
Received by a
non-failing
Node must result
in a response.
Partition – Tolerance
No set of failures
less than total network
Failure is allowed to
Cause the system to
Respond incorrectly.
12