INESCTEC

Edge Compute at Hyperscale

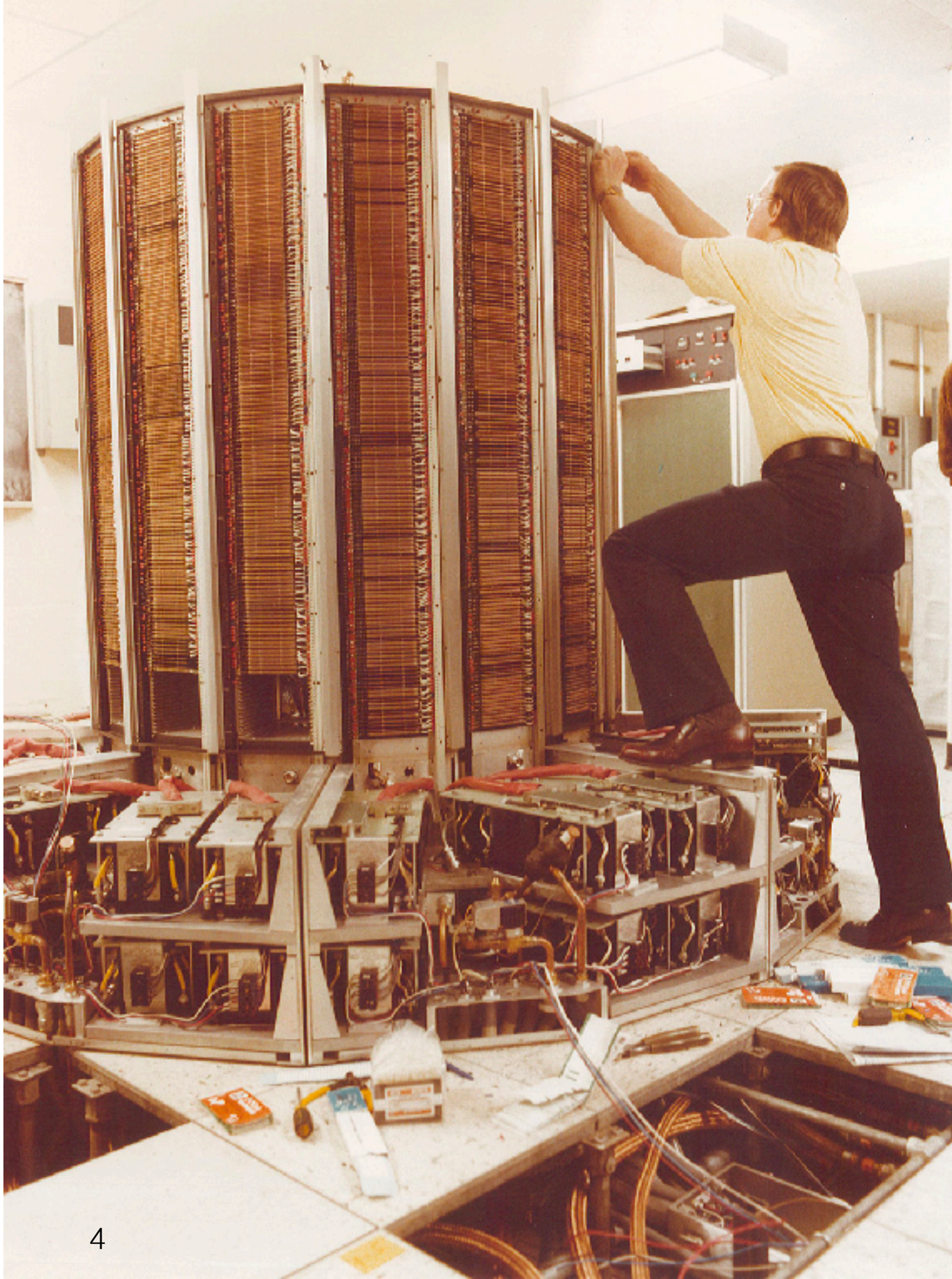
Ali Shoker —INESC TEC & Minho univ., Portugal
Stockholm, 2018

LIGHTKONE
Lightweight computation for networks at the edge


Norwegian Sea
Iceland
Sweden
Finland
Norway
Estonia
Moscow
Москва
Latvia
North Sea
Lithuania
United
Kingdom
Denmark
Belarus
Ireland
Poland
London
Netherlands
Berlin
Ukraine
Belgium
Paris
Czechia
Slovakia
Moldova
Austria
Hungary
Romania
Black Sea
France
Croatia
Serbia
Braga
Bulgaria
Istanbul
Turkey
Italy
Barcelona
Madrid
Rome
Spain
Tyrrhenian Sea
Greece
Lebanon

To scale is to scale out

Cray-1 Supercomputer (at EPFL), 1975



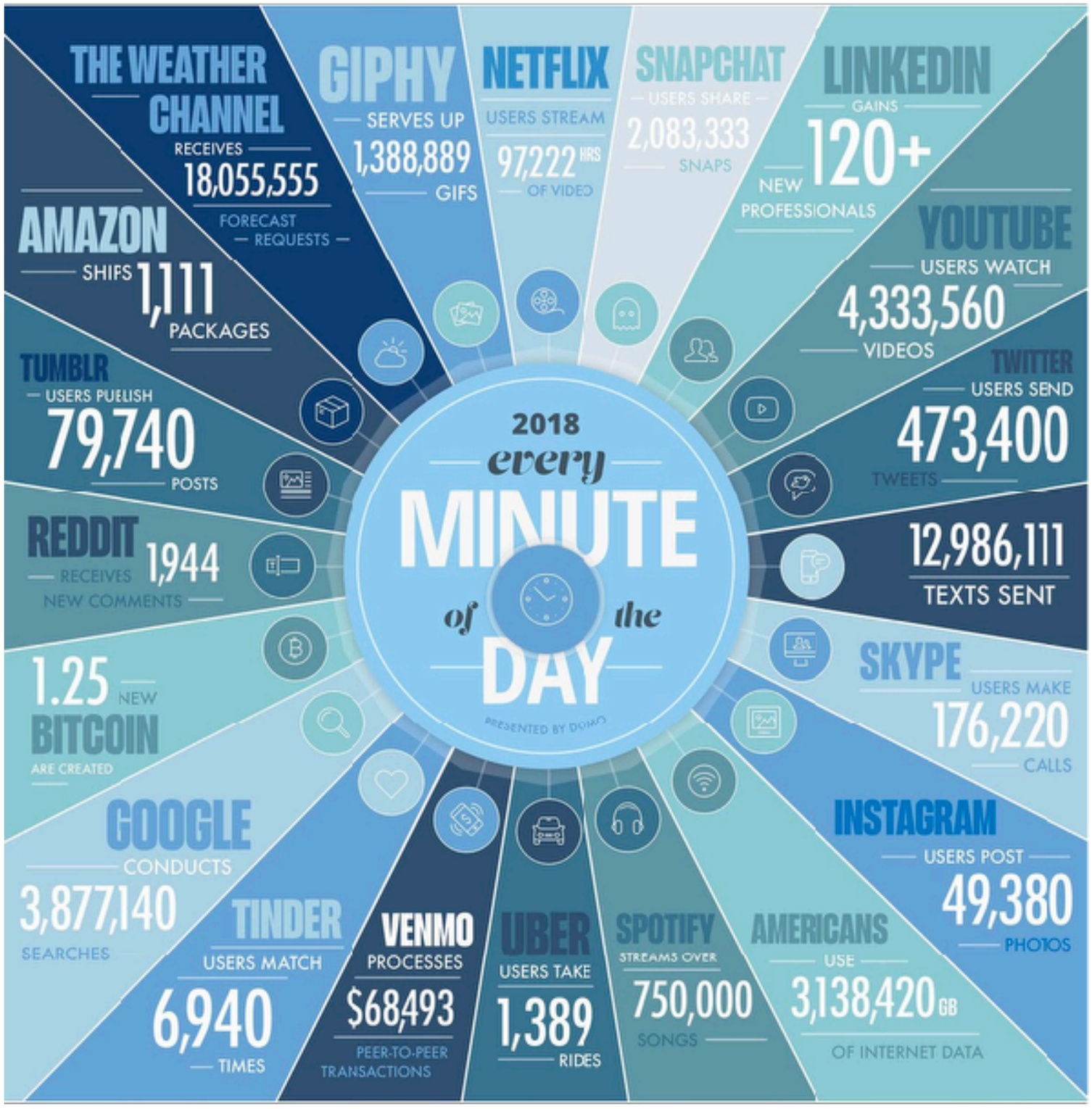
Grid/Cluster computing

90s

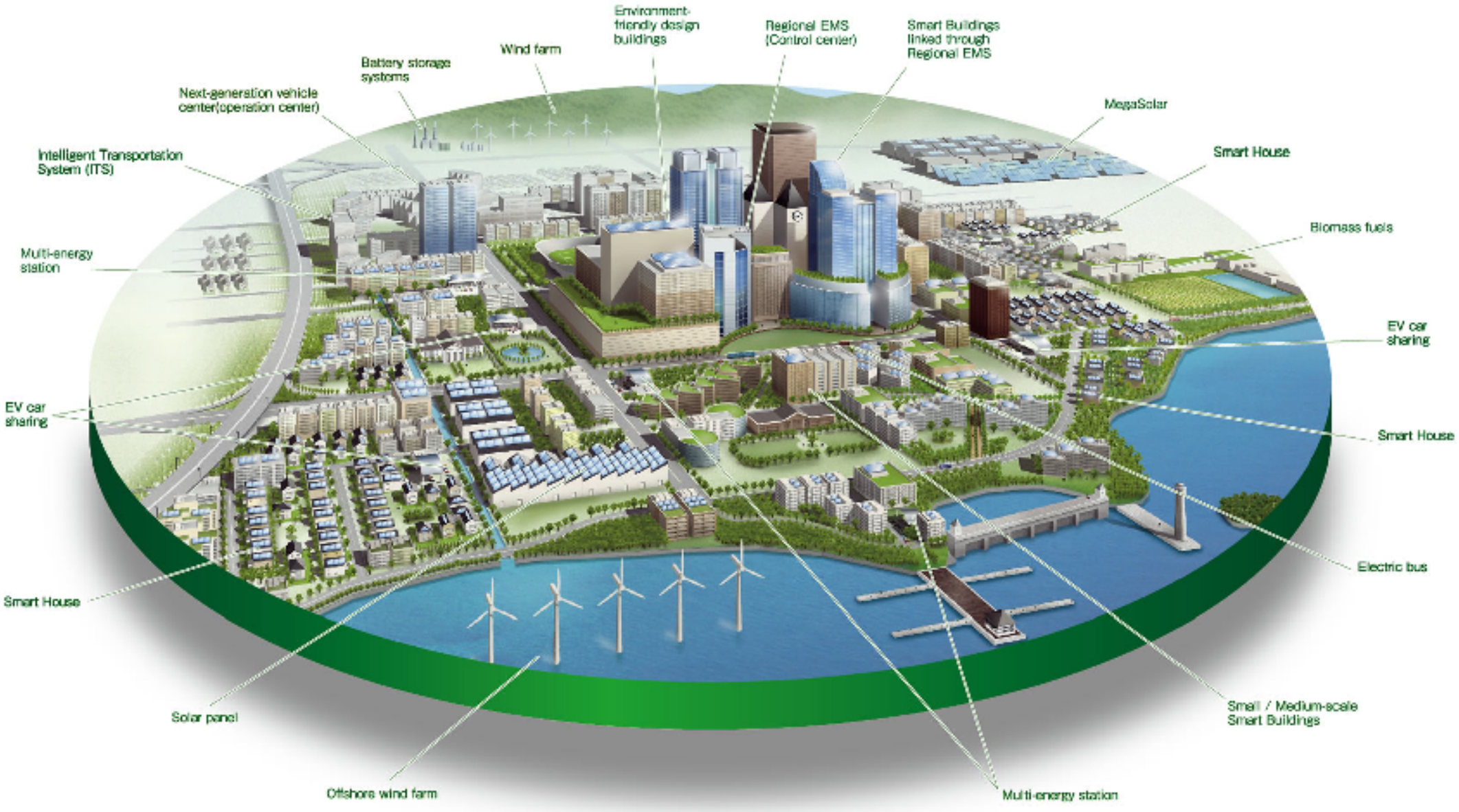
Cloud computing

2000

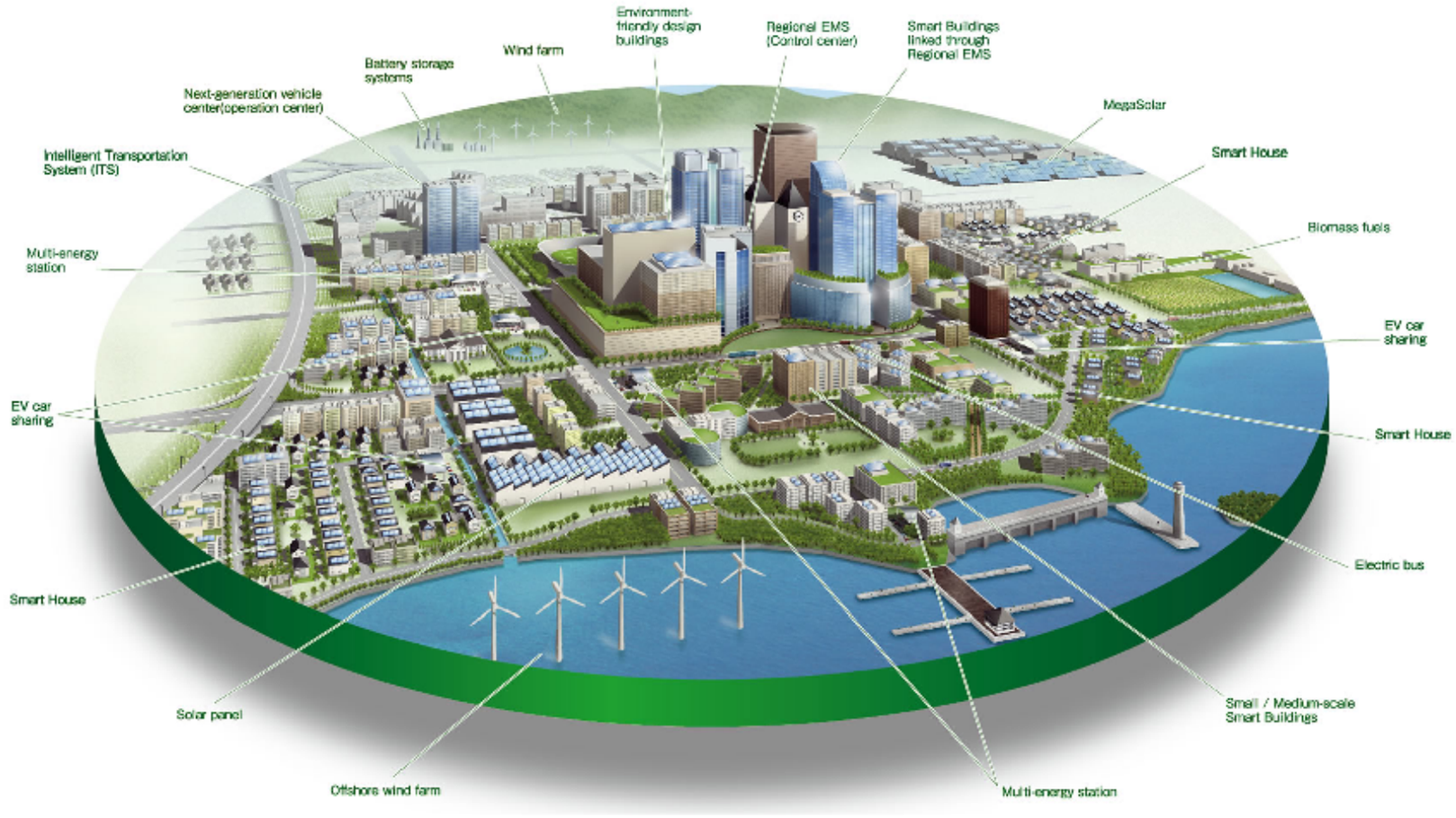
Cloud Overflow



Edge Computing—distributed clouds?



Edge Computing—distributed clouds?



Advantages

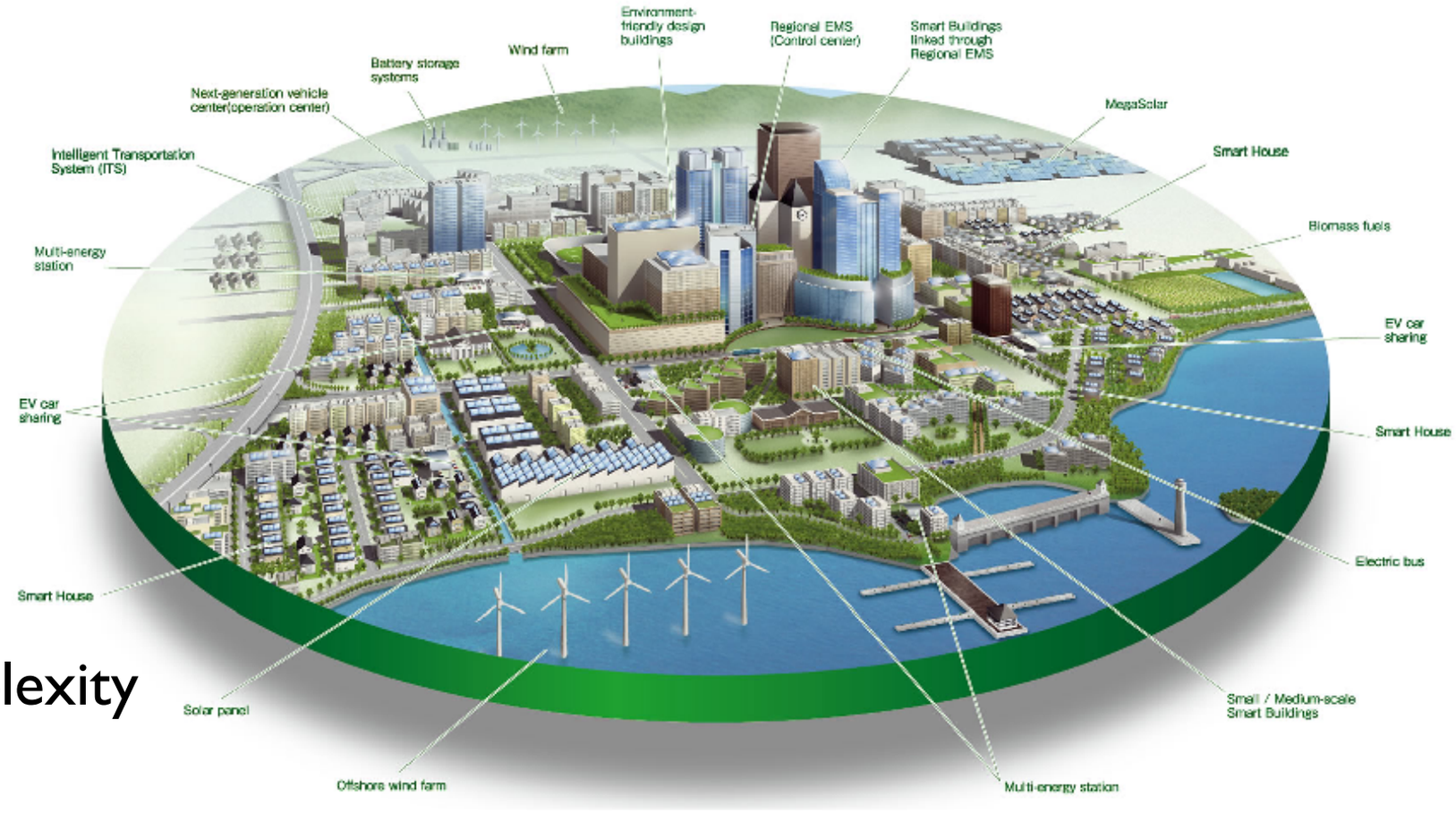
Avoid bottlenecks

Low latency

Save bandwidth

Privacy

Edge Computing—distributed clouds?



Disadvantages

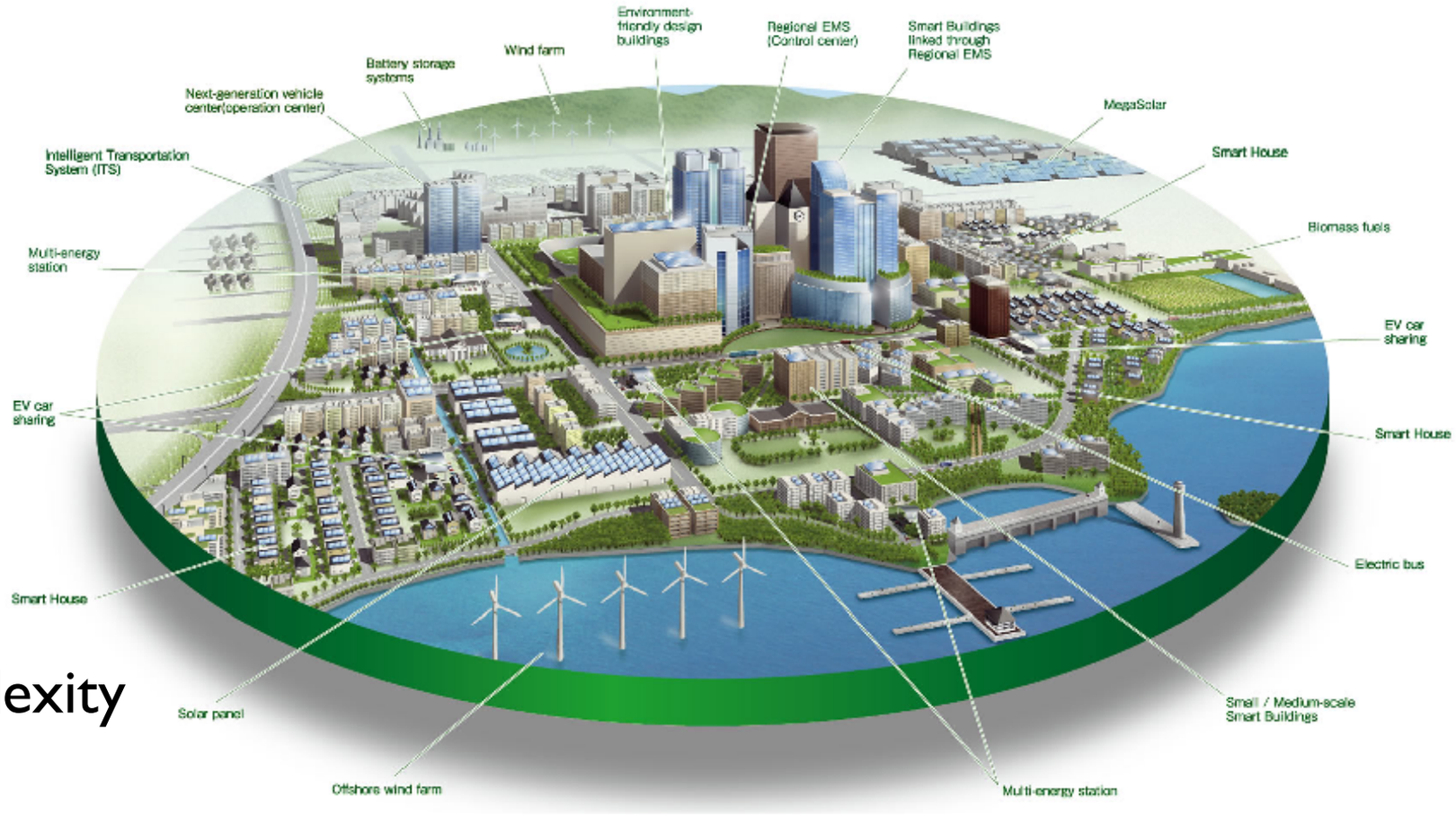
Distributed data complexity

Dynamics

Fragile networks

Heterogeneity…

Edge Computing—distributed clouds?



Disadvantages

Distributed data complexity

Dynamically

Fragile networks

Heterogeneity…

How to build applications?

We focus on data for Edge Computing

Roadmap

How to scale out data

Conflict-free replicated DataTypes (CRDTs)

Hyperscale CRDTs

Outreach and conclusions

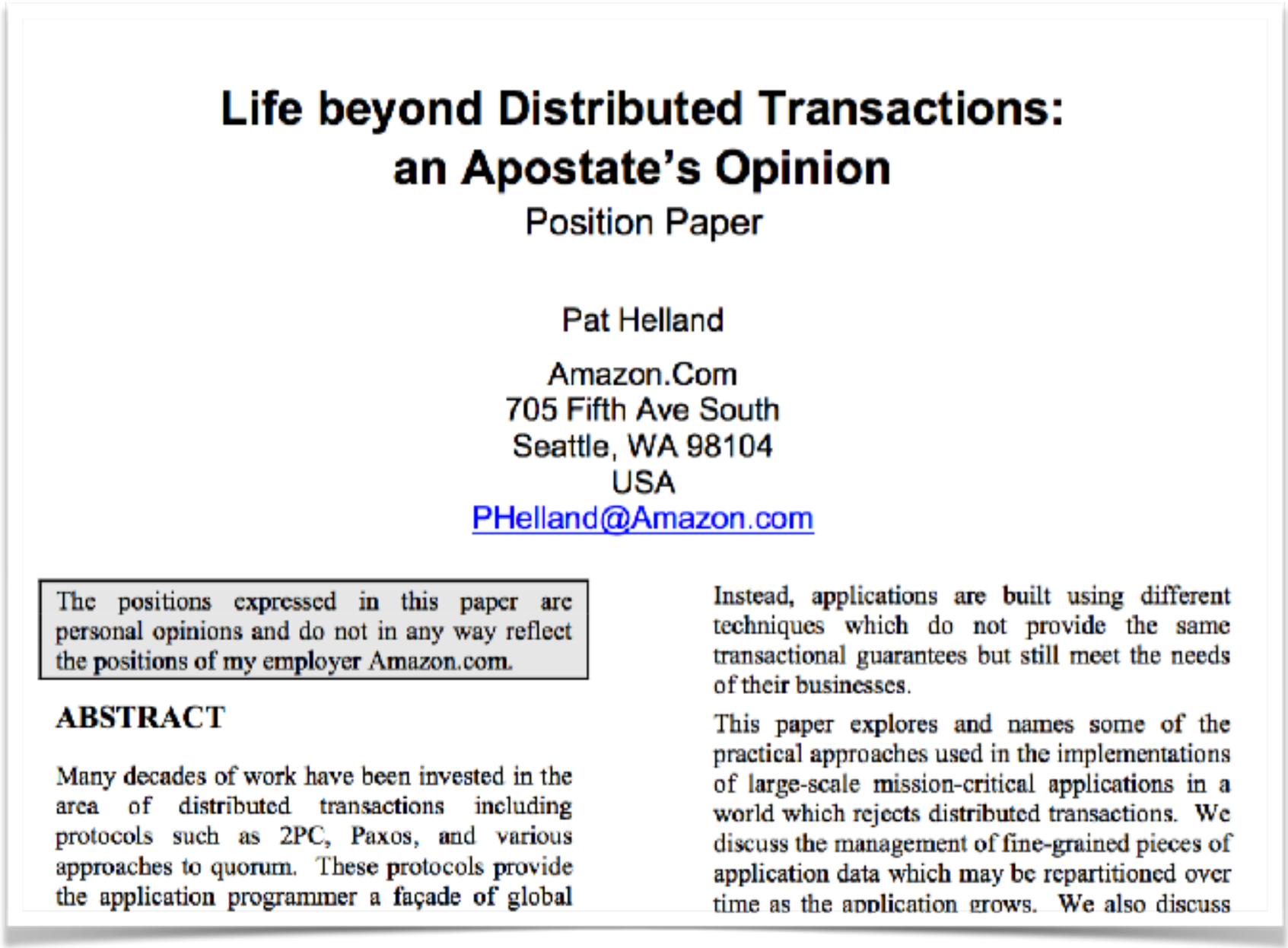
How to scale out data?

Partial replication

Full replication

Almost infinite scalability

partition data in **independent entities** with separate scope of serialisability



Life beyond Distributed Transactions: an Apostate's Opinion

Position Paper

Pat Helland

Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

ABSTRACT

Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC, Paxos, and various approaches to quorum. These protocols provide the application programmer a façade of global

Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

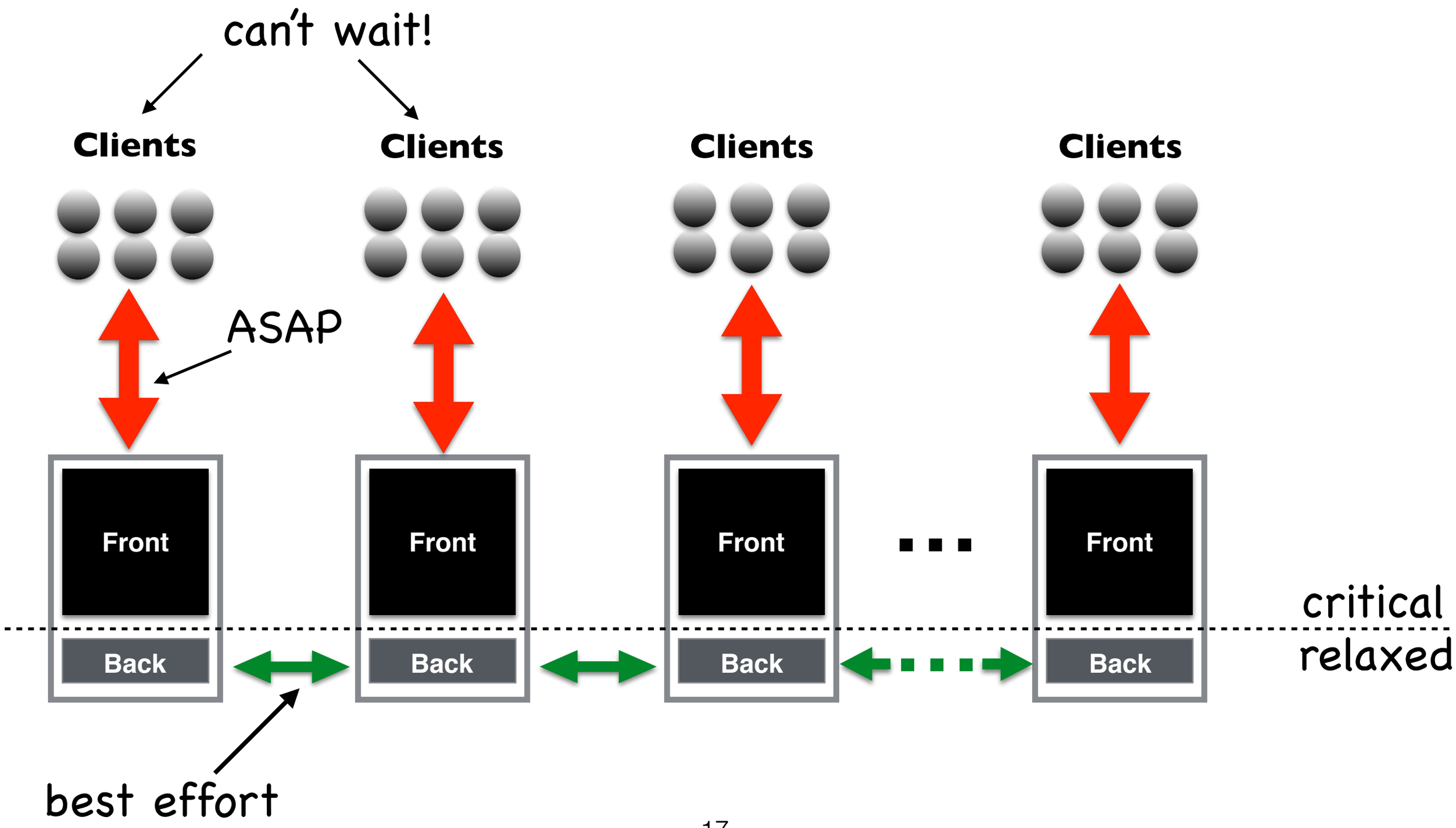
This paper explores and names some of the practical approaches used in the implementations of large-scale mission-critical applications in a world which rejects distributed transactions. We discuss the management of fine-grained pieces of application data which may be repartitioned over time as the application grows. We also discuss

Full replication

CAP theorem: impossible to guarantee all the three:
Consistency, **A**vailability and **P**artition Tolerance?

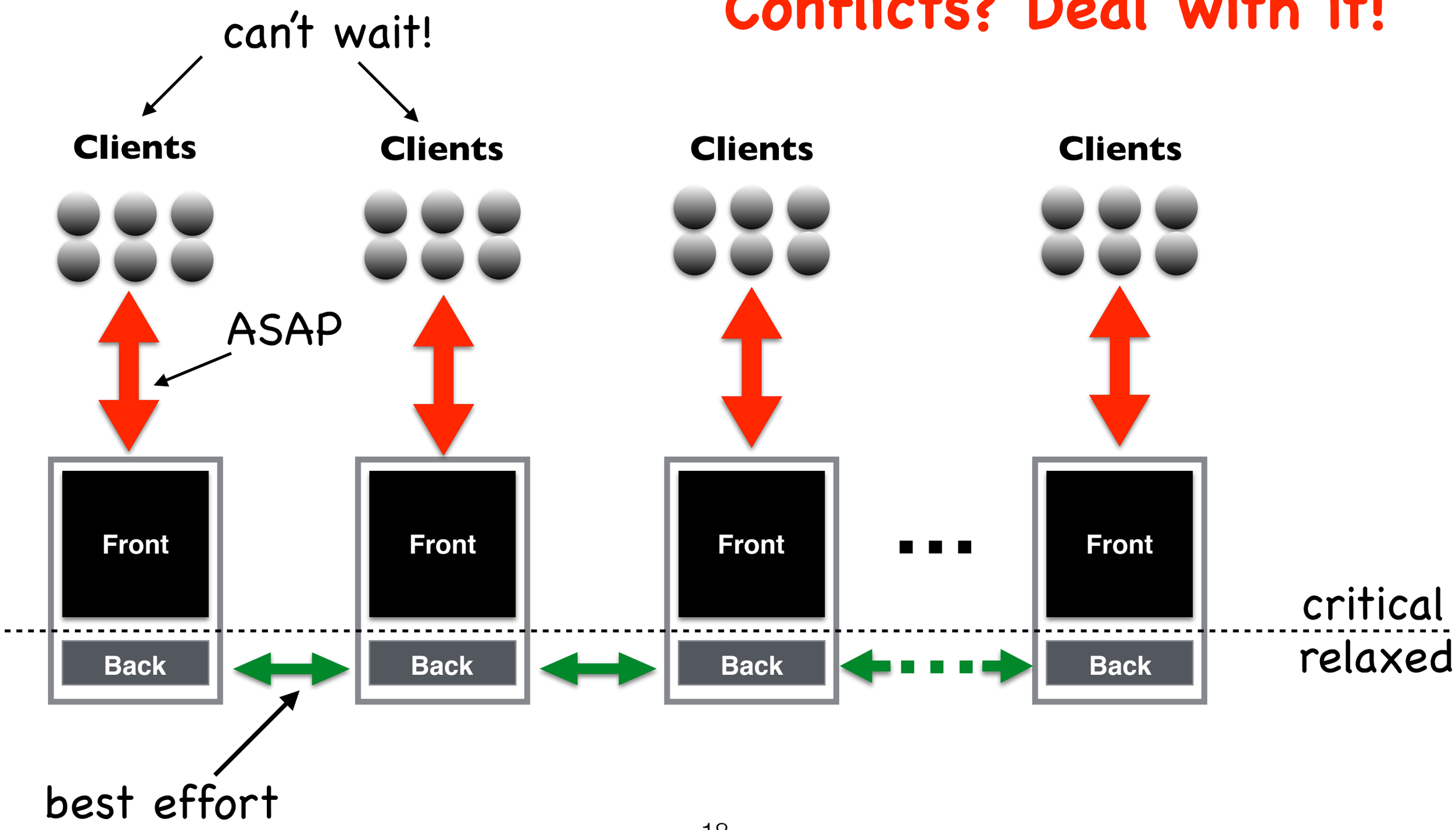
Maybe use **relaxed consistency**?

Relaxed consistency



Relaxed consistency

Conflicts? Deal with it!



Conflict-free **R**eplicated **D**ata**T**ypes (CRDT)

What are they?

Data Types: counters, sets, maps, registers, sequences, ...
Replicated: fully or partially (relaxed consistency)
Conflict-free: mathematically proven to eventually converge

Properties

low sync, commutative operations, internal meta-data

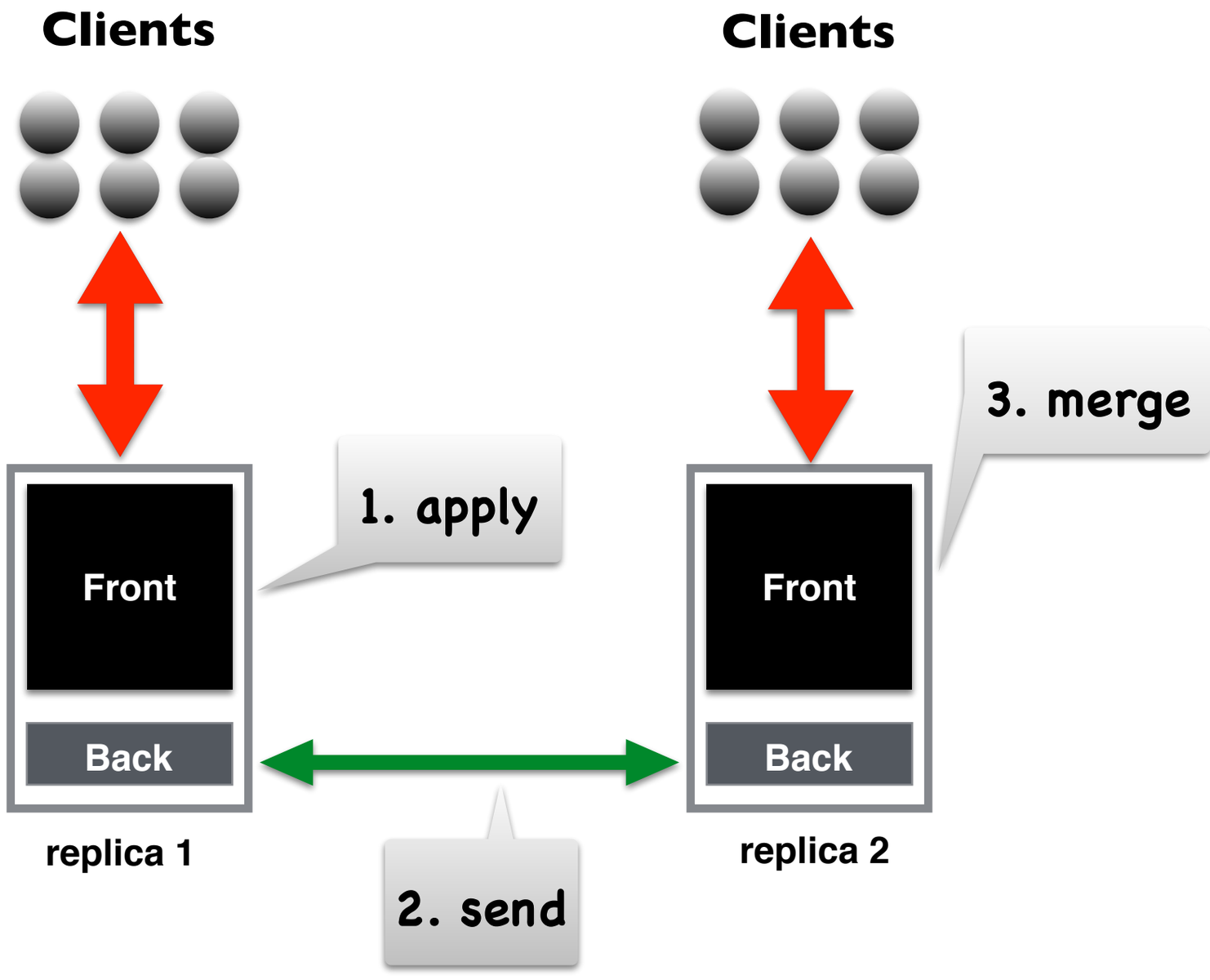
Variants

State-based: disseminate a state
Operation-based: disseminate an operation

State-based CRDT

Worflow

1. apply operation
2. send state (in background)
3. merge state remotely

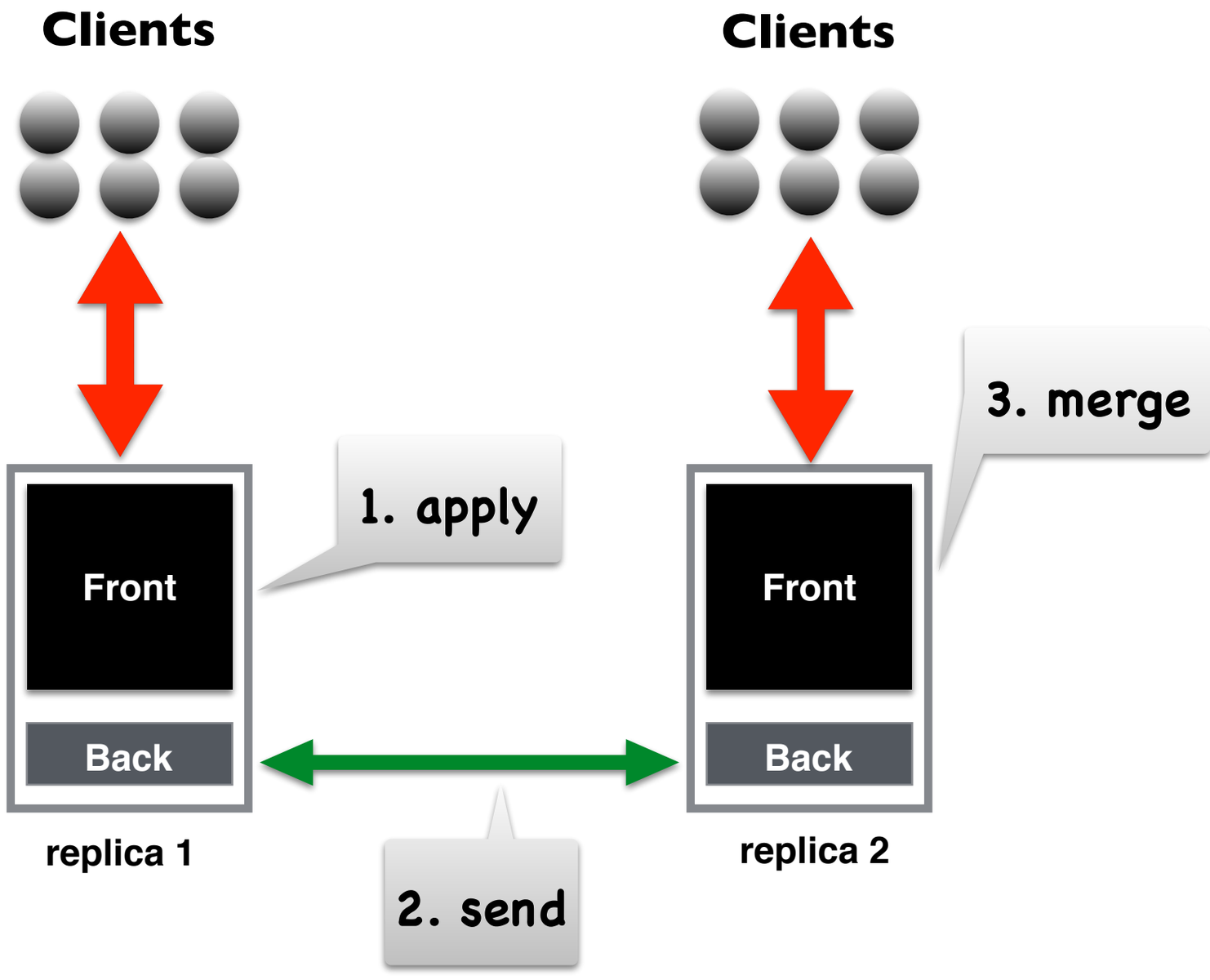


State-based CRDT

Worflow

1. apply operation
2. send state (in background)
3. merge state remotely

inputs:
  $n_i \in \mathscr{P}(\mathbb{I})$, set of neighbors
durable state:
  $X_i := \bot \in S$, CRDT state
on receive$_{j,i}(Y)$
  $X_i' = X_i \sqcup Y$
on operation$_i(m)$
  $X_i' = m(X_i)$
periodically // ship state
  $j =$ random(n_i)
  send$_{i,j}(X_i)$



State-based CRDT

Σ	: State defined as *join semi-lattice*, σ_i is an instance
Mutators	: mutating operations that inflate the state.
$s \sqcup s'$	: LUB to merge states s and s'
$\mathsf{val}_i(q, \sigma_i)$	: Read-only evaluation of query q on a state

State-based CRDT

Σ	: State defined as *join semi-lattice*, σ_i is an instance
$Mutators$	: mutating operations that inflate the state.
$s \sqcup s'$	: LUB to merge states s and s'
$\mathsf{val}_i(q, \sigma_i)$	: Read-only evaluation of query q on a state

- Marc Shapiro, Nuno Preguiça, Carlos Baquero, Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. [Research Report] RR-7506, Inria – Centre ParisRocquencourt; INRIA. 2011, pp.50.
- Paulo Sérgio Almeida, Ali Shoker, and Carlos Baquero. "Delta state replicated data types". JPDC Elsevier journal, 2018.
- Carlos Baquero, Paulo Sérgio Almeida, and Ali Shoker. "Pure Operation-Based Replicated Data Types". arXiv CoRR . October, 2017.

Grow-only Counter (GCounter) CRDT

A shared counter among nodes: n_1, n_2, and n_3

On n_2:

State: a vector v=(2,0,4)

Increment: only my index 2: v=(2,1,4)

Read: the sum => 2+1+4= 7

Merge: (2,0,6) received from n_3:

Pair-wise-max{(2,1,4),(2,0,6)} -> (2,1,6)

$$
\begin{aligned}
\Sigma &= I \hookrightarrow \mathbb{N} \\
\sigma_i^0 &= \{\} \\
\mathsf{inc}_i(m) &= m\{i \mapsto m(i)+1\} \\
\mathsf{val}_i(m) &= \sum_{r \in \mathbf{dom}(m)} m(r) \\
m \sqcup m' &= \mathbf{max}(m, m')
\end{aligned}
$$

So far so good, but how scalable is it?

So far so good, how scalable is it?

State grows linear with number of replicas

$$
\begin{aligned}
\Sigma &= I \hookrightarrow \mathbb{N} \\
\sigma_i^0 &= \{\} \\
\mathsf{inc}_i(m) &= m\{i \mapsto m(i) + 1\} \\
\mathsf{val}_i(m) &= \sum_{r \in \mathbf{dom}(m)} m(r) \\
m \sqcup m' &= \mathbf{max}(m, m')
\end{aligned}
$$

Almost infinite scalability

partition data in **independent entities** with separate scope of serialisability

Life beyond Distributed Transactions: an Apostate's Opinion

Position Paper

Pat Helland

Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

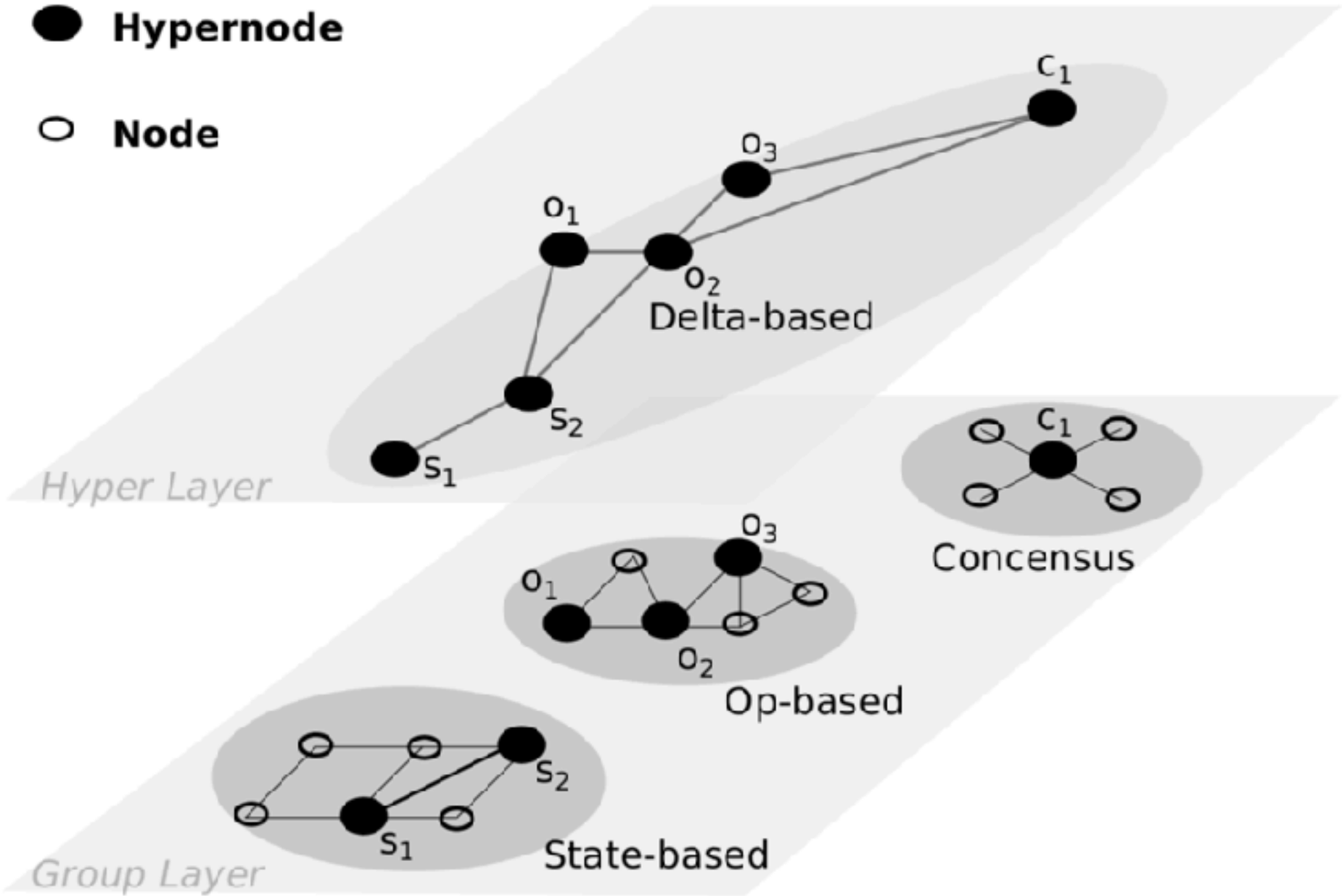
ABSTRACT

Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC, Paxos, and various approaches to quorum. These protocols provide the application programmer a façade of global

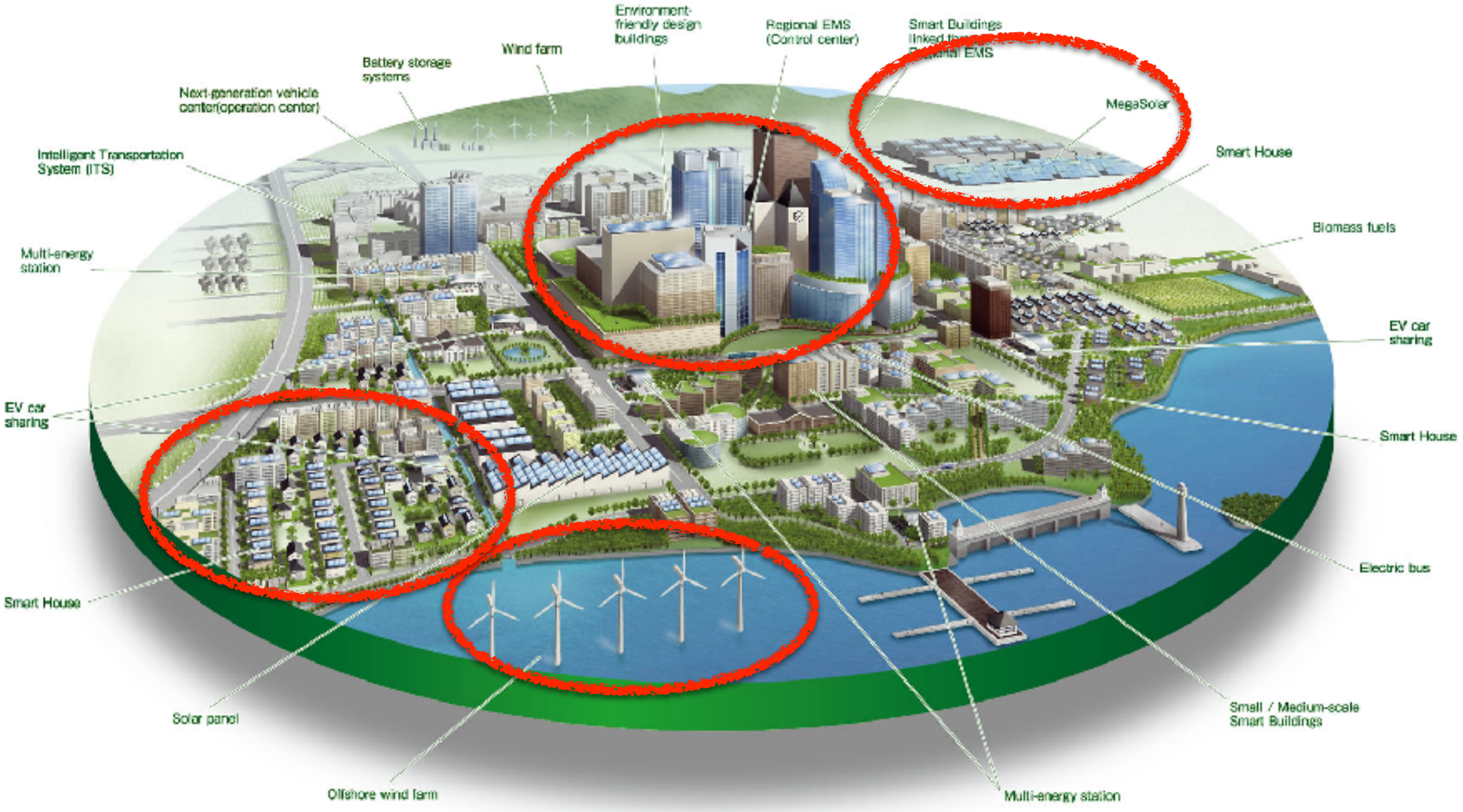
Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

This paper explores and names some of the practical approaches used in the implementations of large-scale mission-critical applications in a world which rejects distributed transactions. We discuss the management of fine-grained pieces of application data which may be repartitioned over time as the application grows. We also discuss

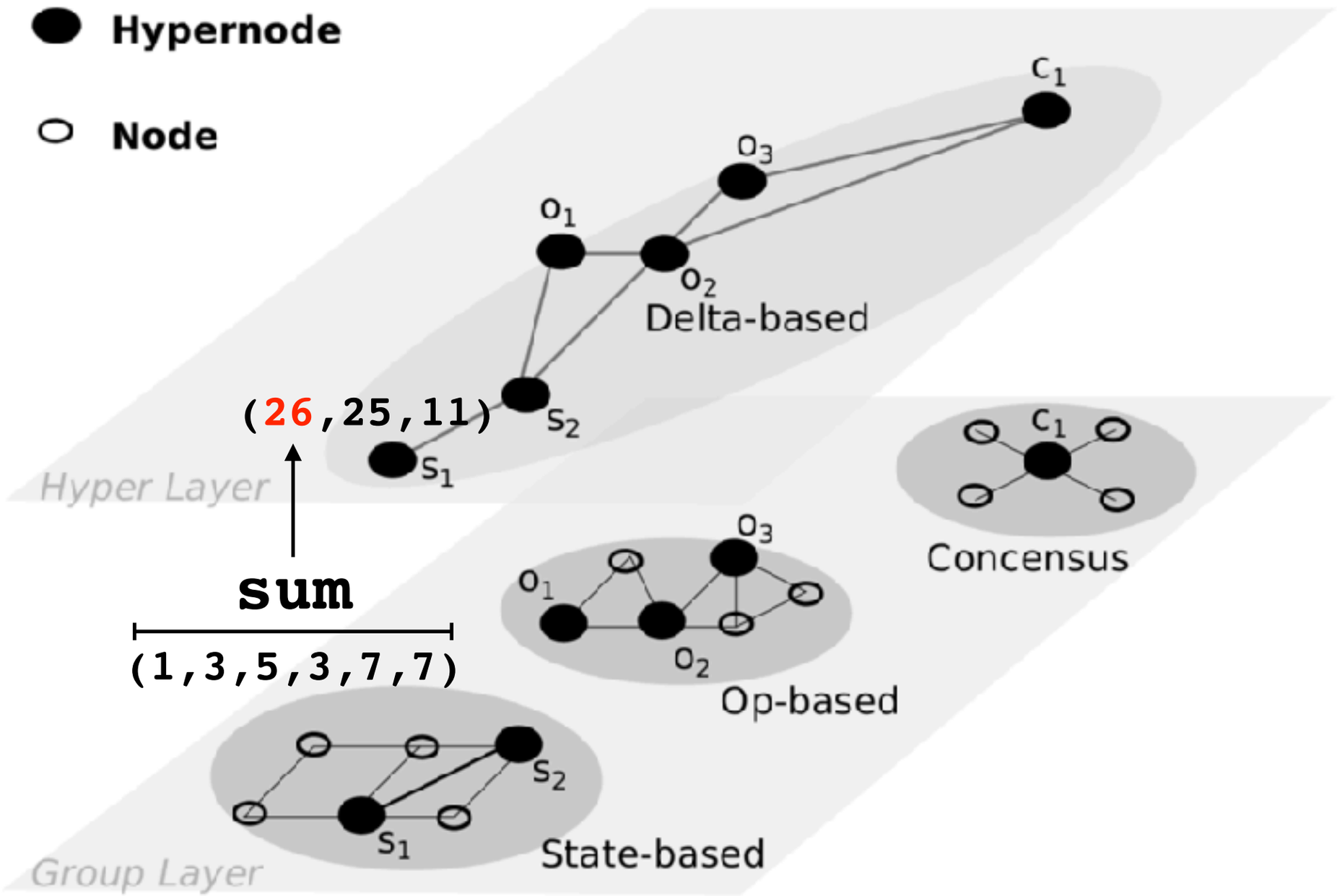
Hyperscale CRDTs



Edge Compute at Hyperscale



Hyperscale GCounter CRDTs



HyperGCounter CRDT

$$
\begin{aligned}
\mathsf{GCDatum}(\mathbb{I}) &= \mathbb{I} \hookrightarrow \mathbb{N} \qquad //\text{a map } m \\
\bot &= \{\} \\
\mathsf{inc}_i(m) &= m\{i \mapsto m(i)+1\} \\
\mathsf{datumvalue}_i(m) &= \sum_{j \in \mathbb{I}} m(j) \\
m \sqcup m' &= pair\_wise\_max(m, m')
\end{aligned}
$$

HyperGCounter CRDT

$$
\begin{aligned}
\mathsf{GCDatum}(\mathbb{I}) &= \mathbb{I} \hookrightarrow \mathbb{N} \qquad \text{//a map } m \\
\bot &= \{\} \\
\mathsf{inc}_i(m) &= m\{i \mapsto m(i)+1\} \\
\mathsf{datumvalue}_i(m) &= \sum_{j \in \mathbb{I}} m(j) \\
m \sqcup m' &= pair\_wise\_max(m, m')
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{GCDigest}(\mathbb{G}) &= \mathbb{G} \hookrightarrow \mathbb{N} \qquad \text{//a map } m \\
\bot &= \{\} \\
\mathsf{update}_k(m, V) &= m\{k \mapsto \max(m(k), V)\} \\
\mathsf{digestvalue}_k(m) &= \sum_{j \in \mathbb{G}} m(j) \\
m \sqcup m' &= pair\_wise\_max(m, m')
\end{aligned}
$$

HyperGCounter CRDT

$$
\begin{aligned}
\mathsf{GCDatum}(\mathbb{I}) &= \mathbb{I} \hookrightarrow \mathbb{N} \qquad //\text{a map } m \\
\bot &= \{\} \\
\mathsf{inc}_i(m) &= m\{i \mapsto m(i)+1\} \\
\mathsf{datumvalue}_i(m) &= \sum_{j \in \mathbb{I}} m(j) \\
m \sqcup m' &= pair\_wise\_max(m, m')
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{GCDigest}(\mathbb{G}) &= \mathbb{G} \hookrightarrow \mathbb{N} \qquad //\text{a map } m \\
\bot &= \{\} \\
\mathsf{update}_k(m, V) &= m\{k \mapsto \max(m(k), V)\} \\
\mathsf{digestvalue}_k(m) &= \sum_{j \in \mathbb{G}} m(j) \\
m \sqcup m' &= pair\_wise\_max(m, m')
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{HyperGC}(\mathbb{L}, \mathbb{G}) &= \mathsf{GCDatum}(\mathbb{L}) \times \mathsf{GCDigest}(\mathbb{G}) \quad //\text{two maps: } l, g \\
\bot &= (\bot, \bot) \\
\mathsf{inc}_i((l, g)) &= (\mathsf{inc}_i(l), \bot) \quad //\text{increment local datum} \\
\mathsf{publish}_{k,}((l, g)) &= (\bot, \mathsf{update}_k(g, \mathsf{datumvalue}(l))) \quad //\text{only hypervisors} \\
\mathsf{hypervalue}_k((l, g)) &= \mathsf{digestvalue}_k(g) \\
(l, g) \sqcup (l', g') &= (l \sqcup l', g \sqcup g')
\end{aligned}
$$

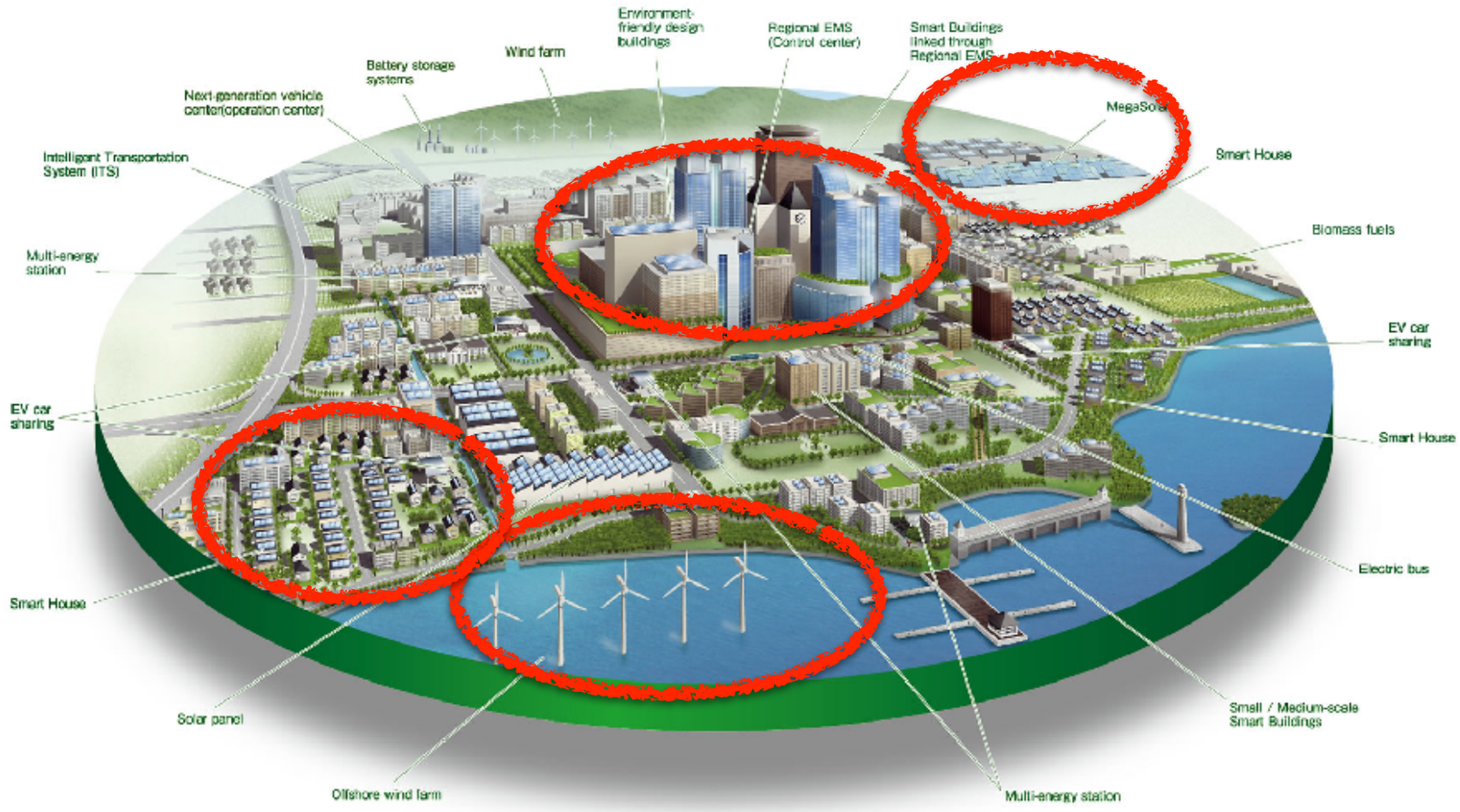
Other Hyper CRDTs

- ❑ GCounter
- ❑ Counter
- ❑ Multi-sets (implemented as maps of counters)
- ❑ Sets (with version-vector metadata)
- ❑ Others

In progress

- ❑ Investigate other CRDTs
- ❑ Allow for remote updates
- ❑ Dynamic membership
- ❑ Empirical evaluations

Concluding remarks



- To scale is to scale out
- Scale out CRDTs: O(N) => O(logN)
- Strength: interoperability, flexibility, storage, bandwidth
- Weekness: complexity, limitations

Get in touch

Ali Shoker
Web: www.alishoker.com
Twitter: ashokerCS