SRECON EUROPE MIDDLE EAST AFRICA

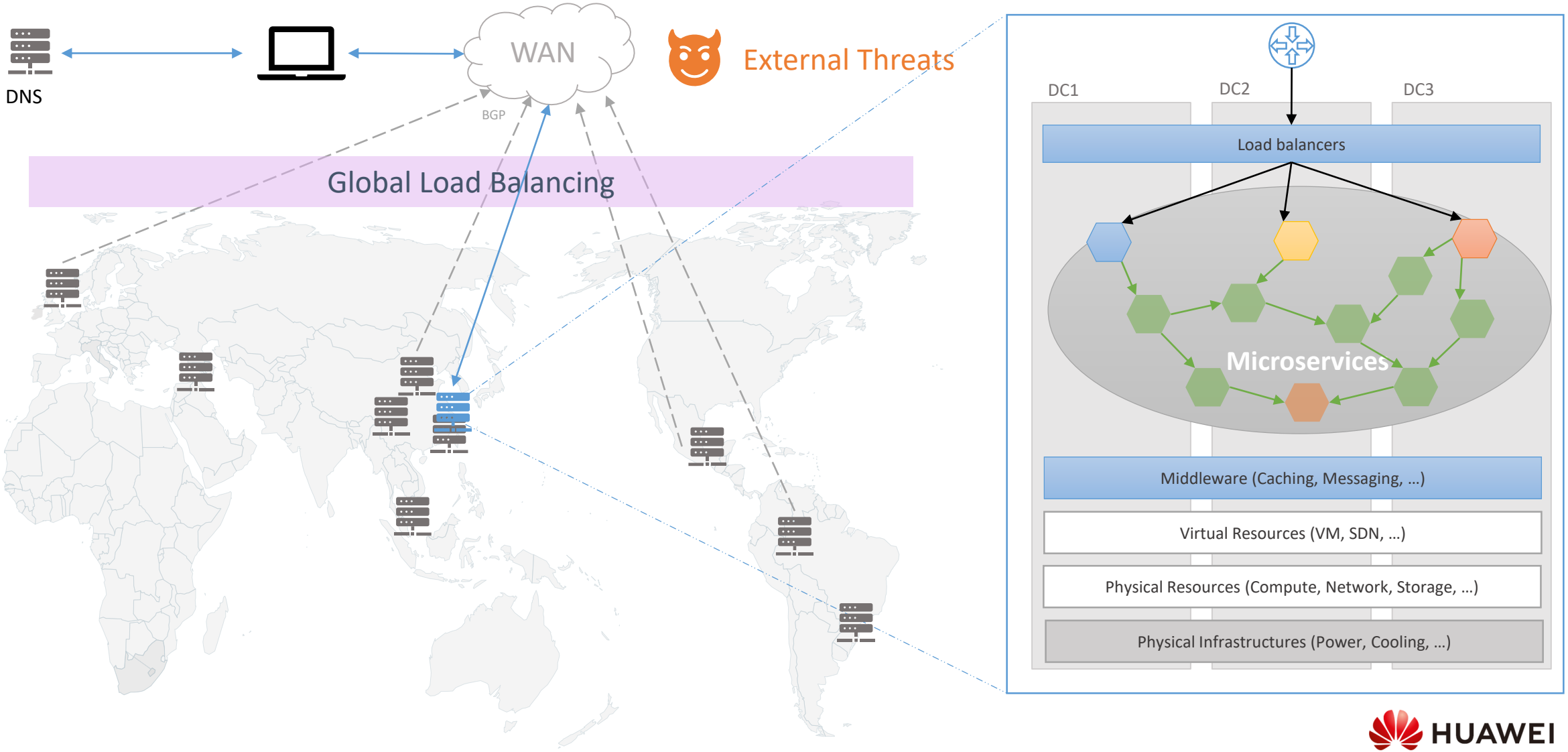
HUAWEI

Blast Radius Reduction

for large-scale distributed systems

Linhua Tang *(aka **James**)*
Software Engineer/Tech Lead
Cloud Reliability Lab, Huawei Ireland Research Center

Building large-scale distributed system is hard



HUAWEI

Why failure is inevitable?

Charles Perrow *argues that the conventional engineering approach to ensuring safety — building in more warnings and safeguards — fails because* ***systems complexity makes failures inevitable***

- **Black Swan** *(unknown unknowns, known unknowns)* :
  - The bad ones
  - Out of our control and hard to predict
  - Severe impact
  - e.g. natural disasters, network cable cut-off by construction
- **Blind spots** *(known unknowns, unknown knowns)* **:**
  - System errors of human beings
  - Collective work is essential to minimize blind spots
- **Trade-offs** *(known knowns)* :
  - Business trade-offs, cost vs profit
  - Technical trade-offs, such as CAP theorem
  - Relative cost of perfect system is similar to the relative mass of an object nearing light speed

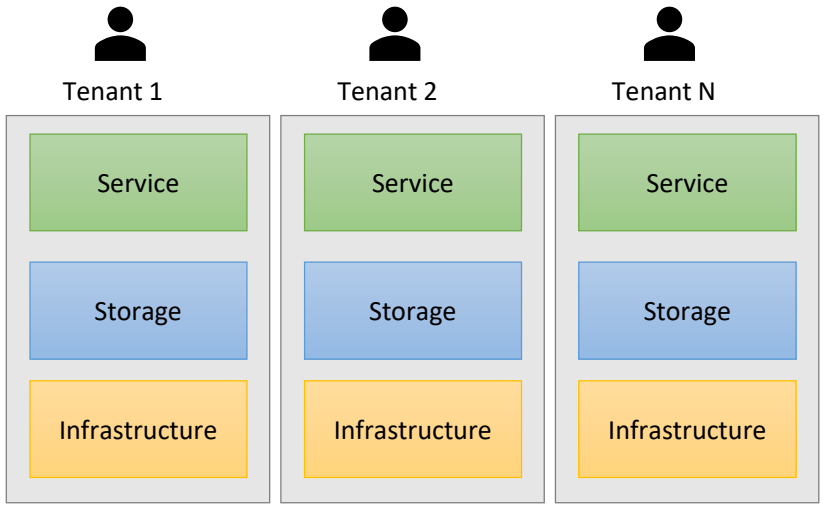
$$m_{\mathrm{rel}} = \frac{m}{\sqrt{1 - \frac{v^2}{c^2}}}.$$

Design for failures: blast radius reduction

- Definition of blast radius per service
  - Tenants
  - Resources
  - Combined or any other workload
- Blast radius reduction must be part of the system design
- Blast radius as one of the reliability metrics of a system
- Blast radius reasoning:
  - failure modes => worst case impacts

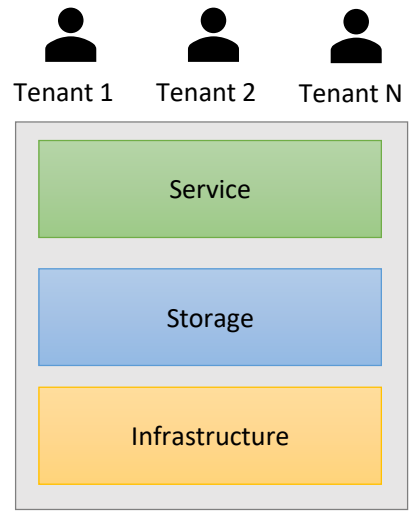
Blast radius reduction with isolation

Isolation is crucial, not optional, for building reliable large-scale distributed systems



Blast radius vs Elasticity (Cost)

- Silo: easy to implement, difficult to maintain and scale, high resource cost
- Full multi-tenancy difficult to implement, highly scalable and cost effective



HUAWEI

Layered Isolation across different services

Throttling/Rate Limit

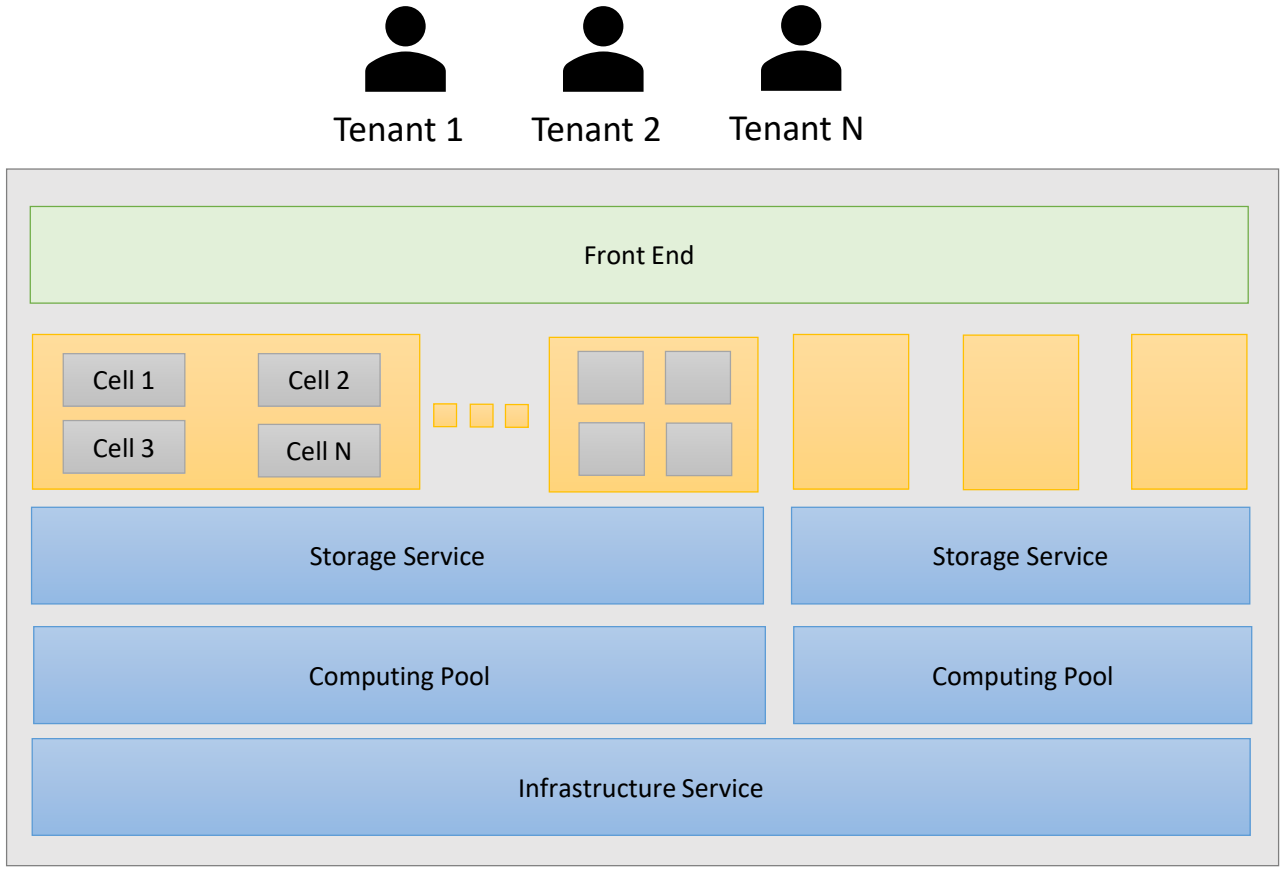
Policy-based isolation

Shuffle sharding

Cell Based Architecture

Software Defined Isolation

Tier-based Isolation



Higher complexity

Strategy Complexity

Lower complexity

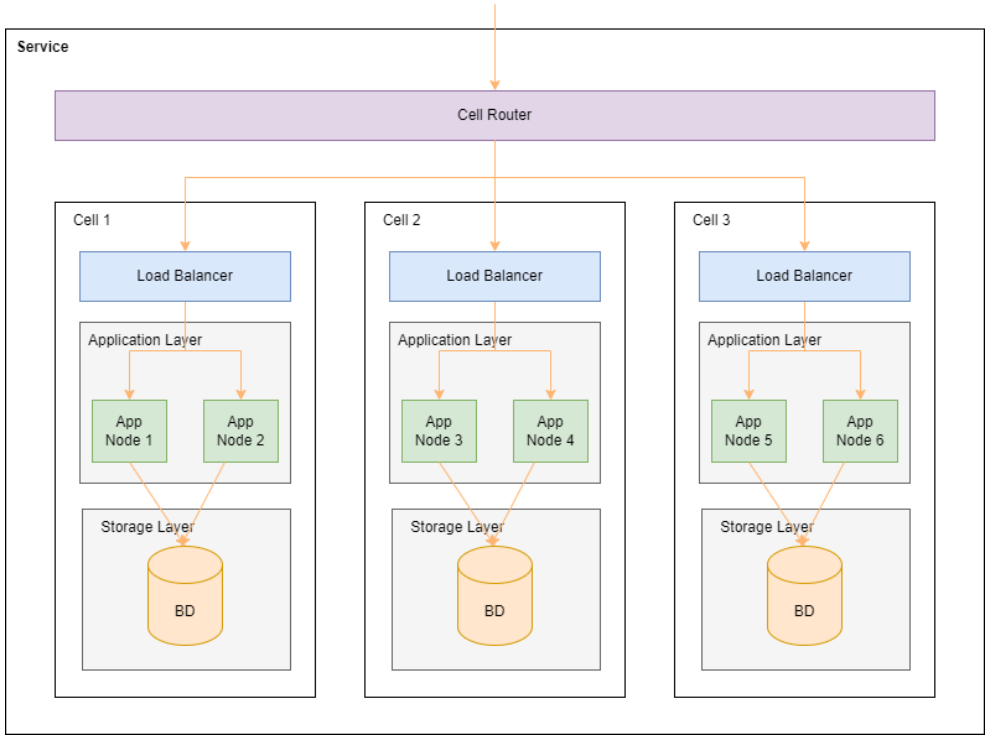
HUAWEI

Cell-based architecture in practice

Cell-based architecture = Fine-grained workload isolation with multiple mini-clusters (hundreds or more), cell is the smallest self-contained scale unit.

Predictable

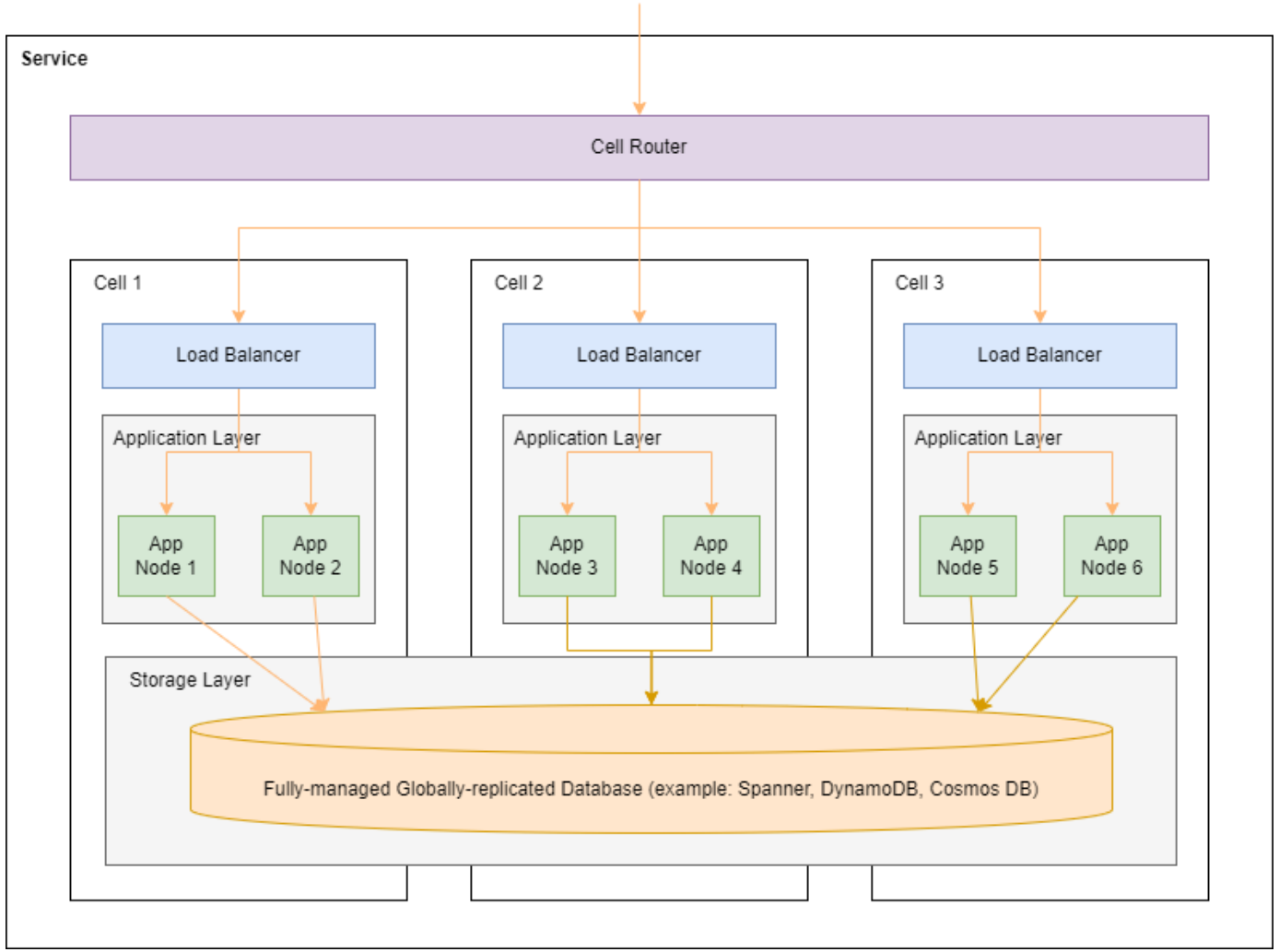
- blast radius
- performance
- scalability



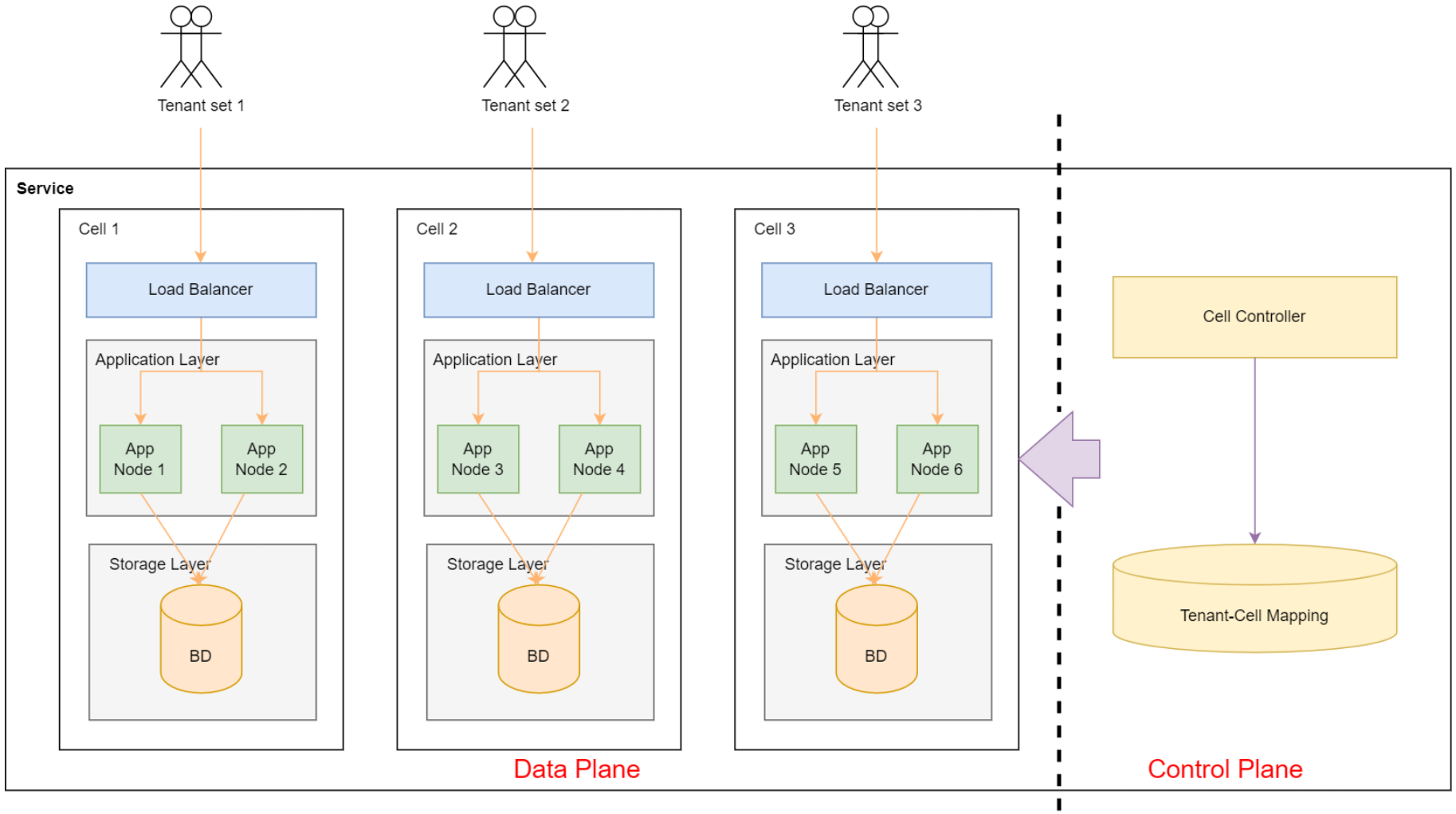
Classic Cell-based architecture patterns

Recommended reference: [AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)]

Cell-based architecture in practice

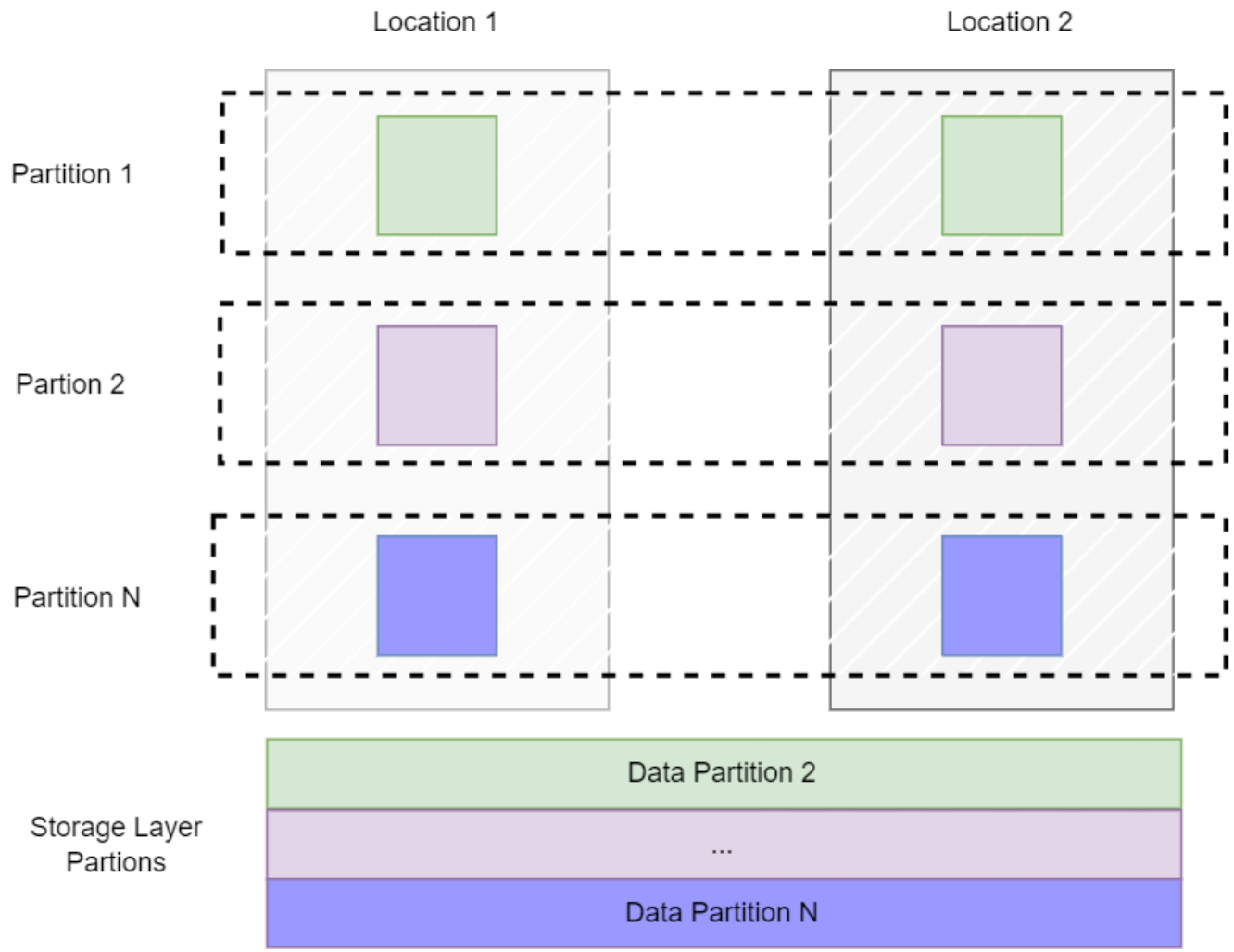


Cell-based architecture in practice

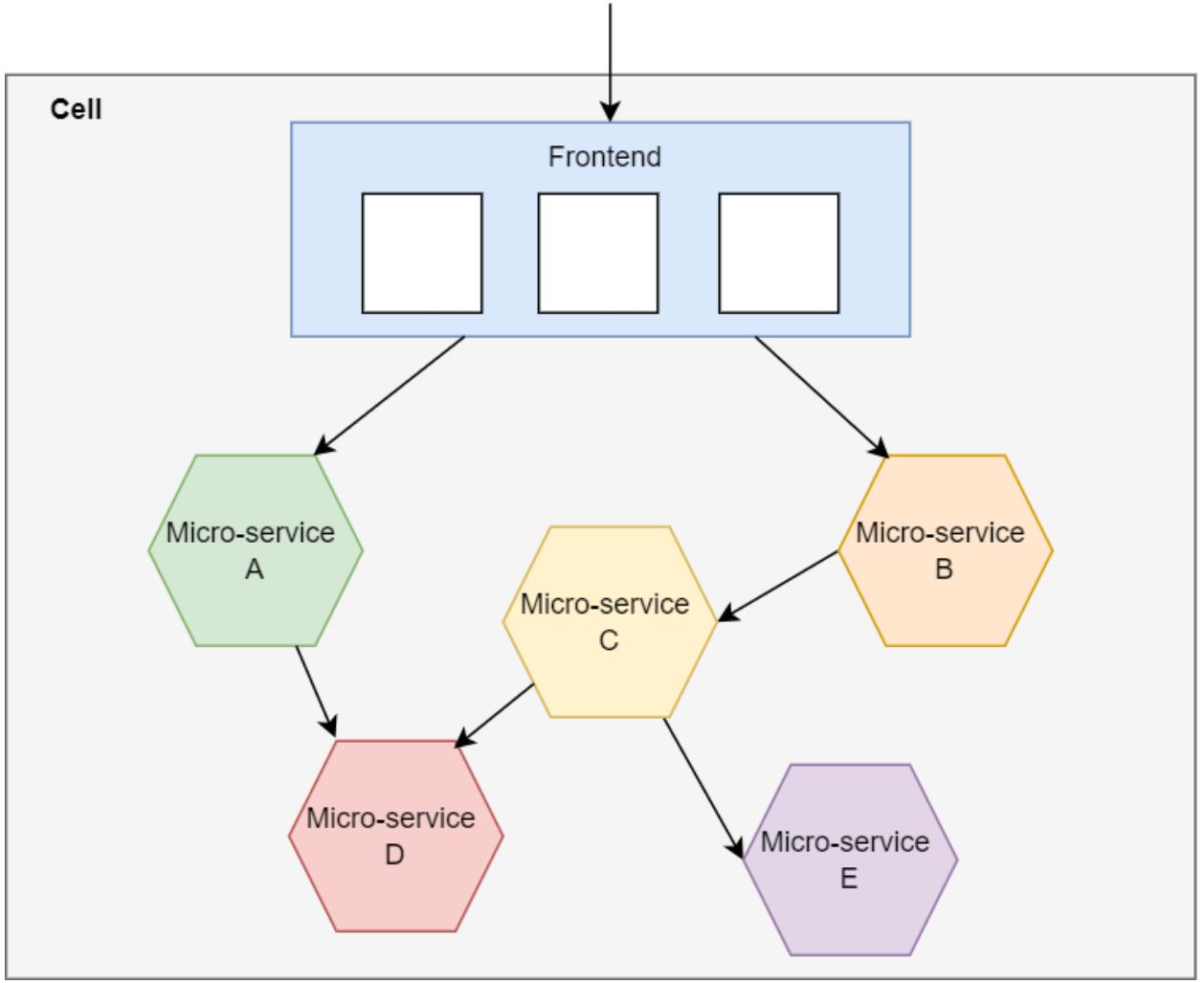


HUAWEI

Cell-based architecture in practice



Cell-based architecture in practice



Cell-based architecture in practice

Challenges and Cost of adopting cell-based architecture:

- Increased **infrastructure cost**: can you split 100GB cache into 1GB per cell?
- Increased **system complexity**: how to migrate customers from one cell to another?
- Increased **operational cost**: how to manage one hundred clusters instead of one (for deployment, monitoring, etc.) ?

XaC (everything as code) and automation is essential for adopting cell-based architecture

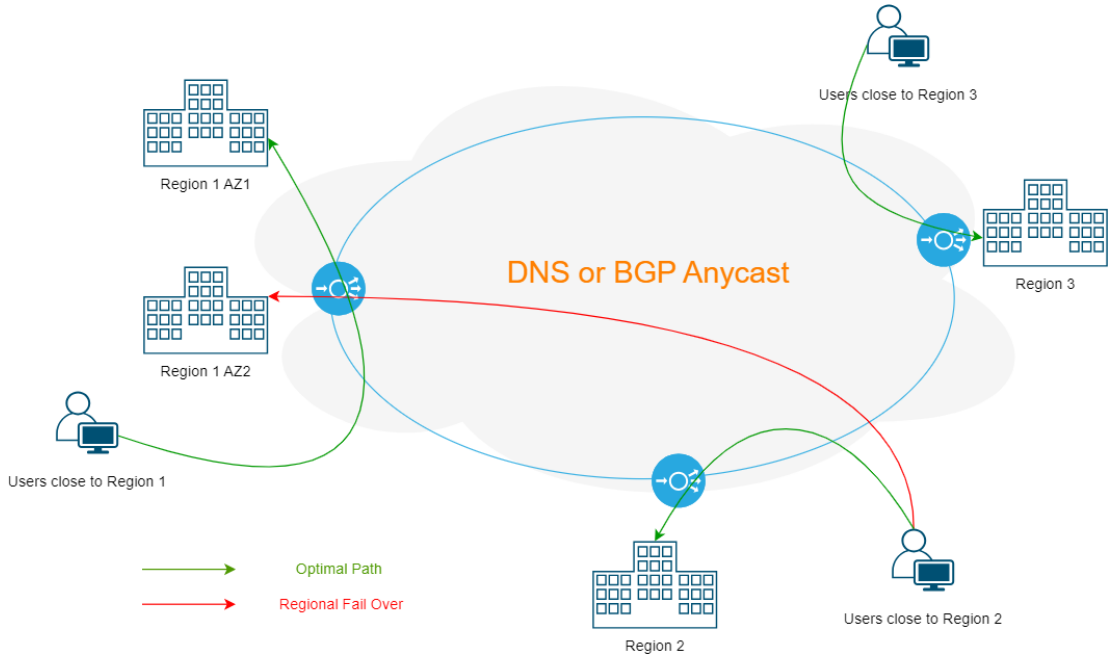
Definitely NOT for small scale systems

Blast radius reduction with self-healing and fast recovery

- Temporal blast radius
  - For example: incident duration 1 hour vs 1 minutes
- Mean time to Recovery (MTTR) is not useful in practice
- Max Time to Recovery (MxTTR) make more sense
- Self-healing and fast recovery is the key to reduce MxTTR
  - **Self-healing**: fully automated recovery mechanism for failures or gray failures, such as physical/virtual machine failures
  - **Fast recovery**: automated or semi-automated with fast detection, fast recovery tools, etc.

Fast recovery with global load balancing

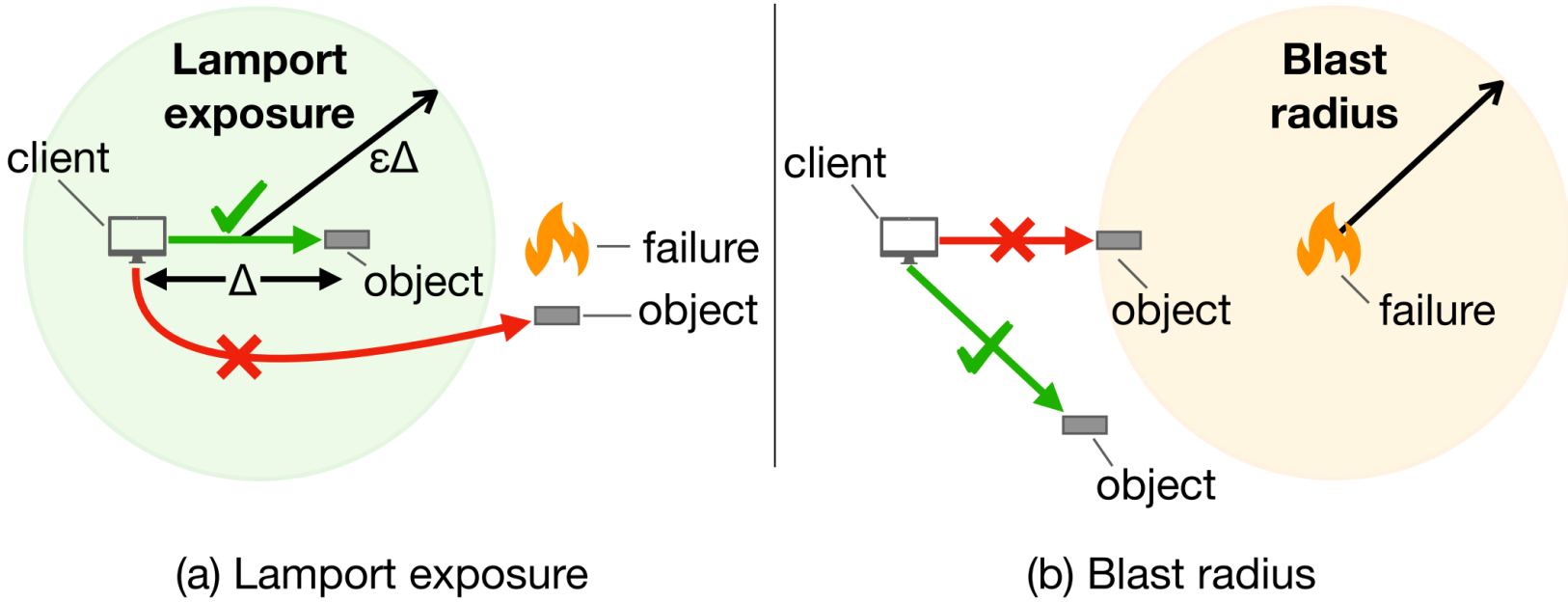
- Global load balancing is a multi-layer load balancing solution
  - DNS
  - Level 4 LB
  - Level 7 LB
- Global traffic steering
  - DNS-based
  - BGP-based



HUAWEI

Beyond blast radius: Lamport Exposure

It proposes that distributed services need not and should not expose local activities to distant failures or partitions, no matter how severe.



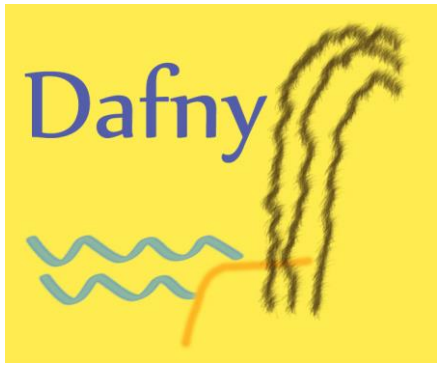
(a) Lamport exposure (b) Blast radius

The diagram from the original paper https://arxiv.org/abs/1405.0637

Băsescu, Cristina, et al. "Limiting Lamport Exposure to Distant Failures in Globally-Managed Distributed Systems." arXiv preprint arXiv:1405.0637 (2014).

Beyond blast radius: formally verify reliability

- Conventional software engineering practices works with limitations
  - system test, chaos engineering, disaster recovery drill, etc.
- Exploring the potential of formal methods to verify the reliability of systems rigorously and exhaustively
  - Formal system specification
  - Model checking
  - Theorem proving



Thank you

- **Linhua Tang** *(aka **James**)*
- https://www.linkedin.com/in/linhuatang/