Tiera / Wiera Design & Architecture

- Tiera handles multi-tiered storage within a single DC
- Wiera handles multiple Tiera instances across DCs

**Motivation**

- **Storage Options in a Datacenter (DC)**
  - Memory (Elasticache, Memcache)
  - Block Device (Cinder, EBS)
  - Object Store (Swift, S3)
- **Public DCs and Edge Nodes**
  - Amazon, Microsoft, Google, …
- **Desirable Combinations**
  - Memcache + S3
  - EBS + Edge nodes’ resource
  - Amazon memory + Azure disk
  - More combinations

Each offers different performance, cost reliability guarantees, and interface

**Finish Line**

- **A system that is aware of the different storage’s**
  - Performance, Durability characteristics
  - Interfaces
  - Cost model
  - Locations (DCs or Edge Nodes)
- **Also knows the application’s**
  - QoS requirements
  - Workload characteristics
  - Consistency model
  - Data durability requirement
  - Fault tolerance requirement

Generates and executes an optimal storage policy that hits the sweet spot in the tradeoff space!

**Using Tiera / Wiera**

- Tiera LowLatencyInstance(time t) {
  % two tiers specified with initial sizes
  tier1: { name: Memory, size: 5G );
  tier2: { name: EBS, size: 5G };

  % Action event (data into memory first)
  event (insert.into) : response {
    insert.object.dirty = true;
    store(what:insert.object, to:tier1);
  }

  %Threshold Event (1GB more memory)
  event(tier1.filled == 75%) : response {
    grow(what: tier1, increment: 1GB);
  }

  %Timer Event (Data copy to disk)
  event(time=t): response {
    copy(what: object.location == tier1,
         to: tier2);
  }
}

- Tiera MultiPrimariesConsistency(time t) {
  % tiers on three datacenter (region)
  region1 = { name: US-West,
             tier1: { name: Memory, size: 5G
                     tier2: { name: EBS, size: 5G };
  region2 = { name: US-EAST,
             tier1: { name: Memory, size: 5G
                     tier2: { name: EBS, size: 5G }};
  region3 = {name: US-CENTRAL,
             tier1: {name: S3, size: 20G}};

  % Action event (replicating data to other DCs)
  event (insert.into) : response {
    lock(what:insert.key)
    store(what:insert.object, to: local.tier1)
    copy(what:insert.object, to: regions.tier1)
    unlock(what:insert.key)
  }

  %Timer Event (for reduced cost)
  event(time=t): response {
    move(what:cold_object_data, to: region3.tier1);
  }
}

**Challenges**

- **Scalability**
  - Millions of objects
  - Millions of requests per second
- **Metadata Management and System Monitoring**
  - Ensure durability
  - Efficient metadata lookup
  - Minimizing impact on workload
  - Characterize application workload
- **Detecting and Avoiding Conflicting Policy Actions**
  - Back up data while serving requests
  - Dynamic re-configuration without affecting performance
- **Achieving Required Quality of Service**
  - Deal with Multi-tenancy
  - Characterizing cloud storage

**Evaluation**

- **Throughput RD only Workload**
  - MySQL on EBS
  - MySQL on Tiera
  - MySQL on DB

- **Cost of Storage Per GB**
  - More of cloudised storage
  - Performance benefit using near DC memory