GAE Feedback

Distributed Systems 2016-2017

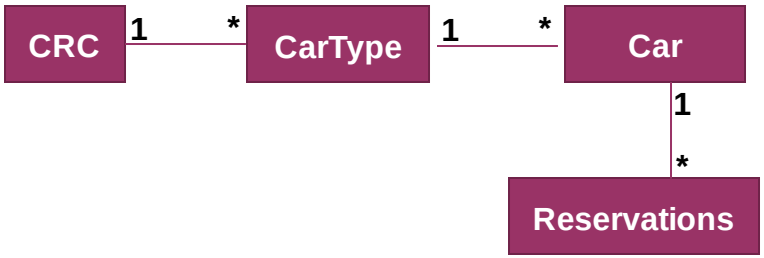
DistriNet
iMinds
KU LEUVEN

Key Attention Points

- 1. Persistence: GAE Datastore using JPA
  - Structure of entities in Entity Groups
  - Use of JPQL
  - Transactions / transactional behaviour
- 2. Indirect Communication
  - Adapt from Direct Communication
  - Use of Task Queues, reason about data flow
- 3. Potential State Inconsistencies (theoretical exercise)
  - Why? How to avoid?
  - How to minimize impact on performance?

1. Persistence

- **Entity Group** (EG): entity relations as hierarchical **tree-structure**
  - one-to-many == parent-child relationship
  - every entity has at max one parent entity (tree)
- Easiest solution in car rental ageny
  - All entities in a single Entity Group
  - No unmanaged relations between those entities
    - E.g. Manual lookup by UID (Primary Key)

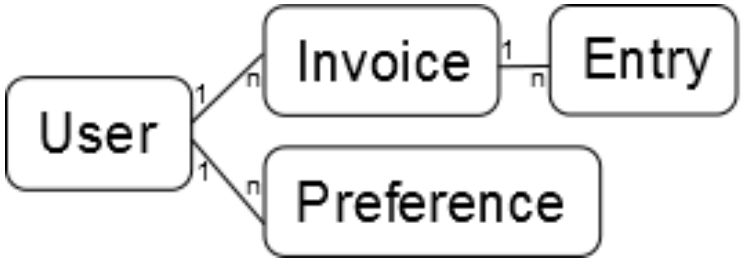


1. Persistence (cont.)

- Google App Engine: JPQL
  - No JOINs supported
  - Use only to query on one entity
- Transactions
  - Each EntityManager session == atomic commit
    - With or without JTA
    - Modification within of a single Entity Group
  - 2 Options for transactional confirming quotes
    - At application level
      - Process all quotes for one CRC within one transaction at a time
      - Undo sucessful reservations if not all quotes could be confirmed
    - At the database level (requires cross-group transactions (XGT))
      - Process all quotes at once
      - Failure to confirm all quotes will trigger automatically rollback

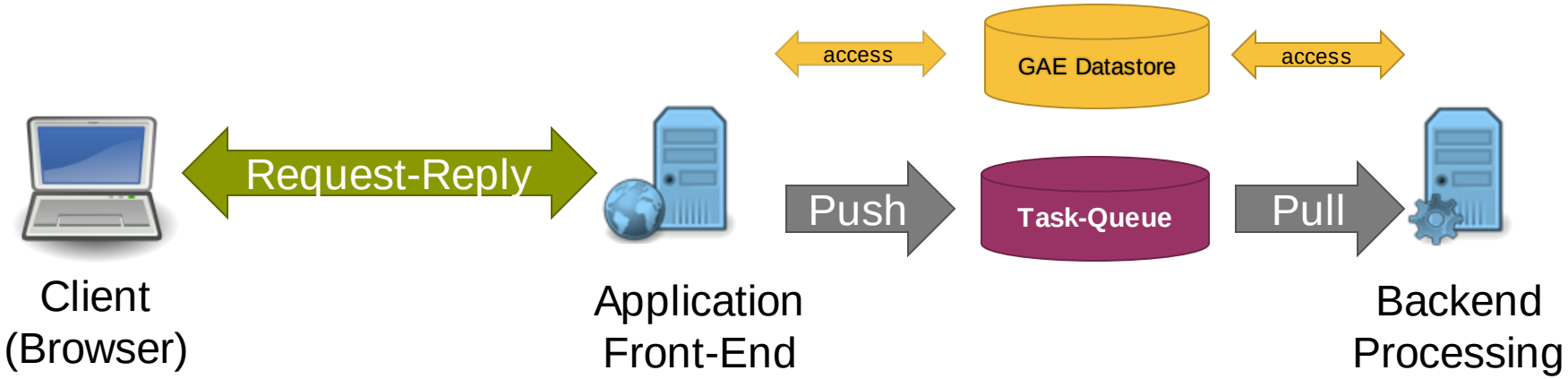
1. Persistence: User Profiles

- User Profiles
  - Independent from car reservations
  - No relation to other profiles
- Entity Group model as depicted in figure
  - Simple modifications will suffice
    - Any change on a single entity group guarantees ACID
    - No need for cross-group transactions



2. Indirect Communication

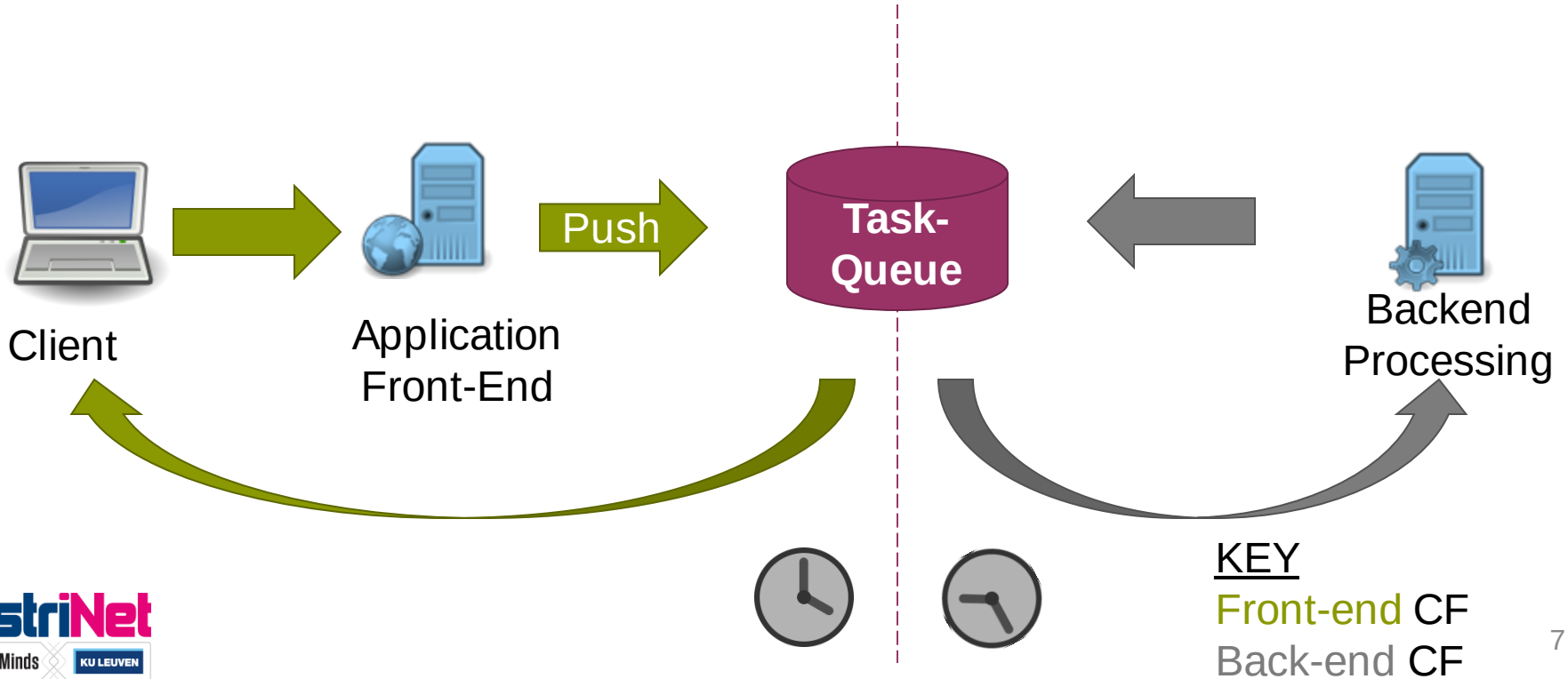
- The **3 roles** involved
  - Data flow:
    - Client (Browser) ← direct → Font-end Service
    - Font-end Service – via Queue → Back-end Service (Worker)



KEY
Direct Communication
Indirect Communication

2. Indirect Communication

- …
  - Control flow (CF):
    - Front-end service triggered by Client
    - Back-end service triggered by GAE infrastructure
    - (1) and (2) are **independent in time**



2. Indirect Communication

- Feedback Channel: Do not
  - Send msgs directly from back-end to front-end
    - client may be gone at time of message
  - wait with front-end until back-end completes
- Examples
  - no shared memory / cache (shared „global“ variables)
  - no Channel API
- Backend workers use separate maschines
  - in own process (⇔ own thread)
  - no shared memory
    - shared „global“ static variables > won‘t work

2. Indirect Communication

- **Feedback channel** from Worker to Client
  - report about success AND failure
  - otherwise indistinguishable : not-yet-processed or failed

- Client may have **multiple** unconfirmed **quotes waiting**, or may have sent a rental order twice.
  - feedback should uniquely refer to quotes (e.g. using **orderID**)

3. Potential State Inconsistencies

- **State changes** (creation of reservations) happen only at the Worker role and not **at the front-end**.
  - Keyword ‚**synchronized**‘ for quote-enqueue function **no effect** on potential state inconsistencies.
- Multiple workers process tasks queue **in parallel** (default)
  - This is where potential inconsistencies are rooted
- Each worker is a separate **process** (separate JVM)
  - thread-based monitors (i.e. ‚**synchronized**‘) **cannot help**

3. Potential State Inconsistencies

- Bogus behavior (double booking) possible?
- With Cross-Group Transactions (XGT)
  - No, locking at the database-level
- Without XGT
  - Depends on GAE implementation of optimistic concurrency control
    - At persistence layer: no, concurrency control sufficient
    - At middleware layer: yes, different processes not aware of each other >> solution addresses this case

3. Potential State Inconsistencies

- One solution
  - Set #worker per queue = 1
  - To increase parallelism
    - Maintain one queue per company
      (**Note**: only if client will book from single CRC)

- NOT
  - Create a queue per CarType or per Quote
    - → loss of all-or-nothing semanics
  - Create a queue for every task
    - japodizes state consistency even more (parallelism →∞)

Questions?