Muppet:
Gyors adatok MapReduce stílusú feldolgozása

Muppet: MapReduce-Style Processing of Fast Data

Tartalom

- Bevezető
- MapReduce
- MapUpdate
- Muppet 1.0
- Muppet 2.0
- Eredmények

Jelenlegi tendenciák

- Nagy mennyiségű adat → MapReduce
- Folyamatosan érkező adat (gyors adat) → ???

Gyors adat

- Folyamatosan érkezik
- Nagy mennyiség
- Minél hamarabb szeretnénk feldolgozni (low-latency)
- Pl.: Twitter, Facebook, YouTube, Foursquare, Flickr, stb.
- Twitter: napi 400 millió tweet
- Foursquare: napi 5 millió checkin

MapReduce

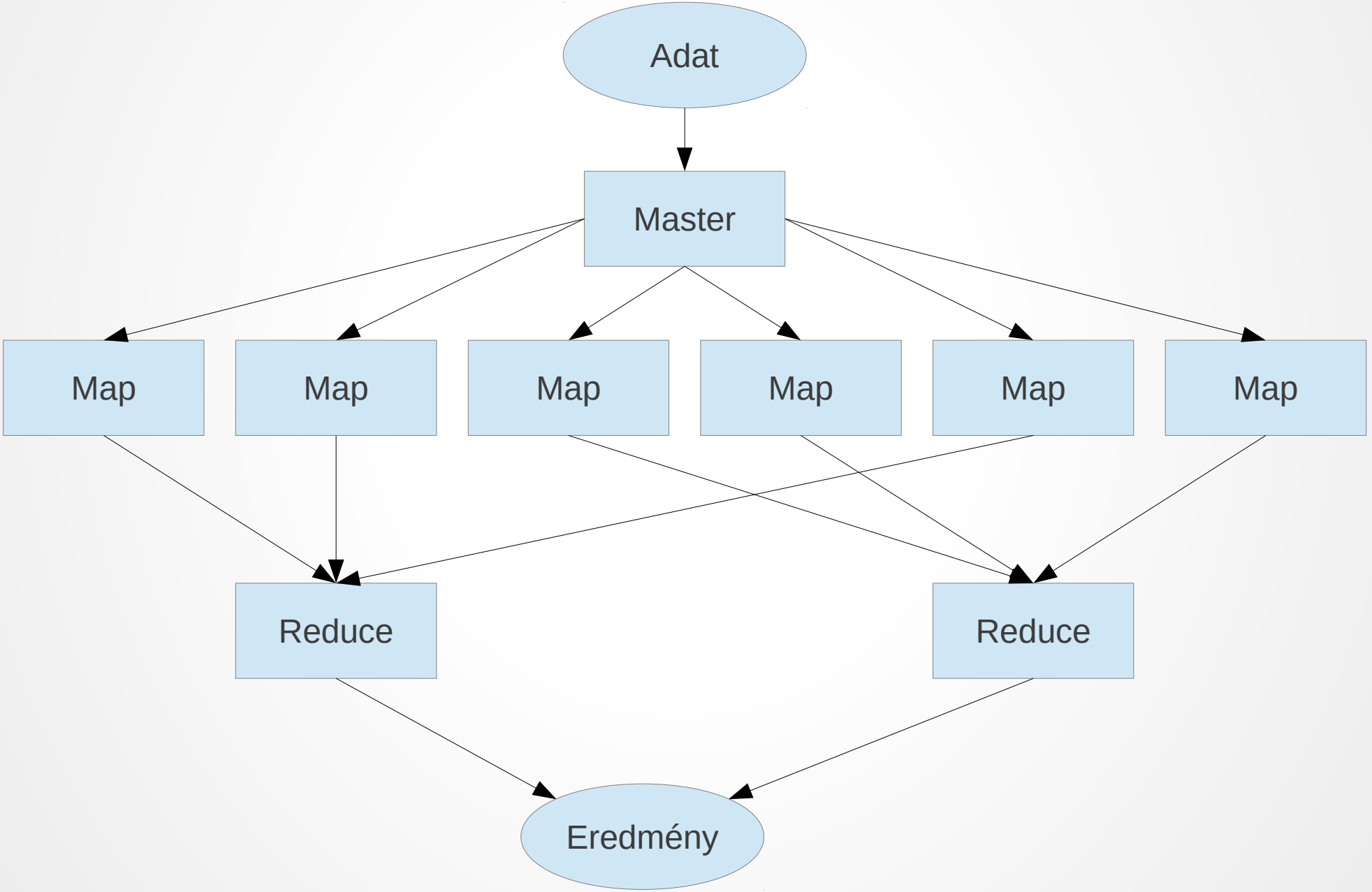
MapReduce

- Programozási modell nagy adathalmaz feldolgozására
- Google
- Elosztott számítás sok gépen
- Alapja map és reduce függvények elosztott végrehajtása
- Sok implementáció, legismertebb: Apache Hadoop

Map és Reduce függvények

- Map(k1,v1) → list(k2,v2)
  - Paraméterek a kezdeti adat kulcsa és tartalma
  - Eredmény elemek listája a reduce számára
- Reduce(k2, list (v2)) → list(v3)
  - Key szerint csoportosítva hívódik meg
  - Azonos kulcsú értékek között számol

MapReduce



MapReduce példák

- Elosztott grep
- Megnyitott URL-ek összegzése
- Fordított web-link gráf
  - egy URL milyen címekről érhető el
- Dokumentum indexelés
  - Bizonyos szavak melyik dokumentumokban fordulnak elő

Hátrányok

- Reduce csak a teljes Map után indulhat
- Új elem esetén újra kell számolni
- Az adat nem változhat a futás közben
- A MapReduce futás elkezdődik és megáll

MapUpdate

Miért nem jó a MapReduce?

- Az adatforrás „streaming”
- A program indításakor nem áll rendelkezésre az összes adat
  - Ezt a felhasználók még nem hozták létre
- A futás során az adat folyamatosan érkezik
- Az adatfolyam soha nem ér véget

MapUpdate

- MapReduce-hoz nagyon hasoló
  - Reduce helyett Update függvény kell
- Szükséges platform is hasonló
  - Több memória és kevesebb diszk kell
- A bejövő adatok alapján valamilyen adatszerkezetet frissítünk a memóriában
  - MapReduce nagy fájlokon dolgozik

Event

- Event <sid, ts, k, v>
  - Stream ID
  - Timestamp
  - Kulcs
    - Nem egyedi az események között
    - Pl. egy Tweet-elő felhasználó ID-ja
  - Érték
    - Tetszőleges blob
    - Pl. egy tweet-et leíró JSON

Stream

- Események sorozata
  - Egy adott stream-en azonos sid
  - Ts szerint rendezett
  - Lehet belső vagy külső
    - Pl. Tweet-ek
    - Pl. map vagy update által generált események

Map

- map(event) → event*
  - Feliratkozik egy vagy több stream-re
  - Ezekről ts szerint rendezetten kapja az eseményeket
  - Egy esemény feldolgozásakor kibocsát 0 vagy több eseményt tetszőleges stream-ekre
  - A kibocsátottak ts-e nagyobb, mint az input
    - Cél stream lehet az is, amire feliratkozott, de nagyobb ts esetén ez nem okoz gondot

Update

- update(event, slate) → event*
  - Input-output map-hez hasonló
  - Kap egy slate-et
    - Memóriában tárolt adatszerkezet
    - Az eddig kapott események alapján ismert adatok
    - Minden (updater, key) párhoz létezik külön slate
    - Folyamatosan frissített
    - Ha a kulcsok Twitter userID-k, a slate tárolhatja pl:
      - Tweet-ek számát
      - Utolsó tweet időpontját
      - Egyéb fontos adatot, melyeket az események alapján aggregált

Update

- update(event, slate) → event*
  - Input-output map-hez hasonló
  - Kap egy slate-et
    - Memóriában tárolt adatszerkezet
    - Az eddig kapott események alapján ismert adatok
    - Minden (updater, key) párhoz létezik külön slate
    - Folyamatosan frissített
    - Ha a kulcsok Twitter userID-k, a slate tárolhatja pl:
      - Tweet-ek számát
      - Utolsó tweet időpontját
      - Egyéb fontos adatot, melyeket az események alapján aggregált

Slate

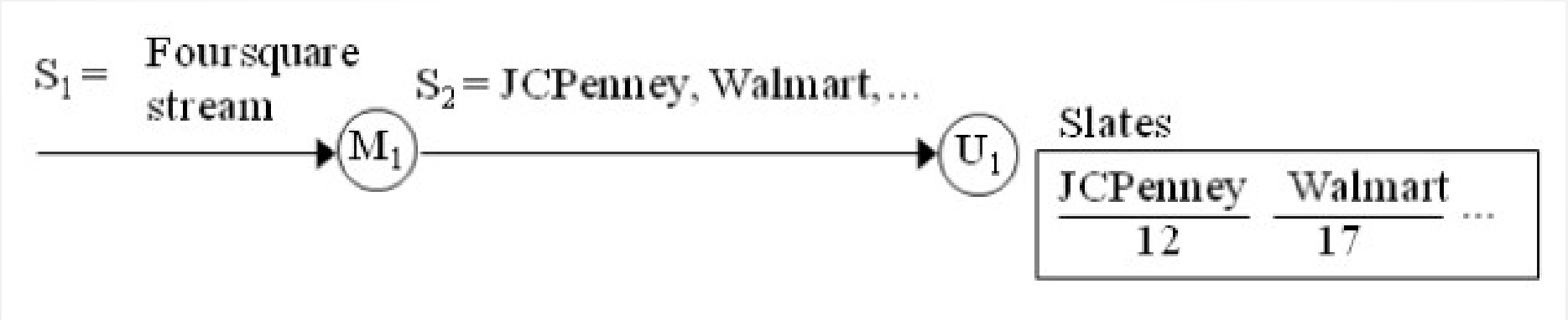
- MapReduce csak az input adatokkal dolgozik
- Slate az update-erek emlékezete
- Lehet time-to-live érték
  - Ha lejár, a slate törlődik (pl. 1 hónapja be nem jelentkezett felhasználók nem érdekesek)
- Ha a slate nem létezik, az updater hozza létre
- Key-value store-ban mentésre kerülnek
  - Ha kell, visszatöltődnek
  - Swap-elés

MapUpdate alkalmazás

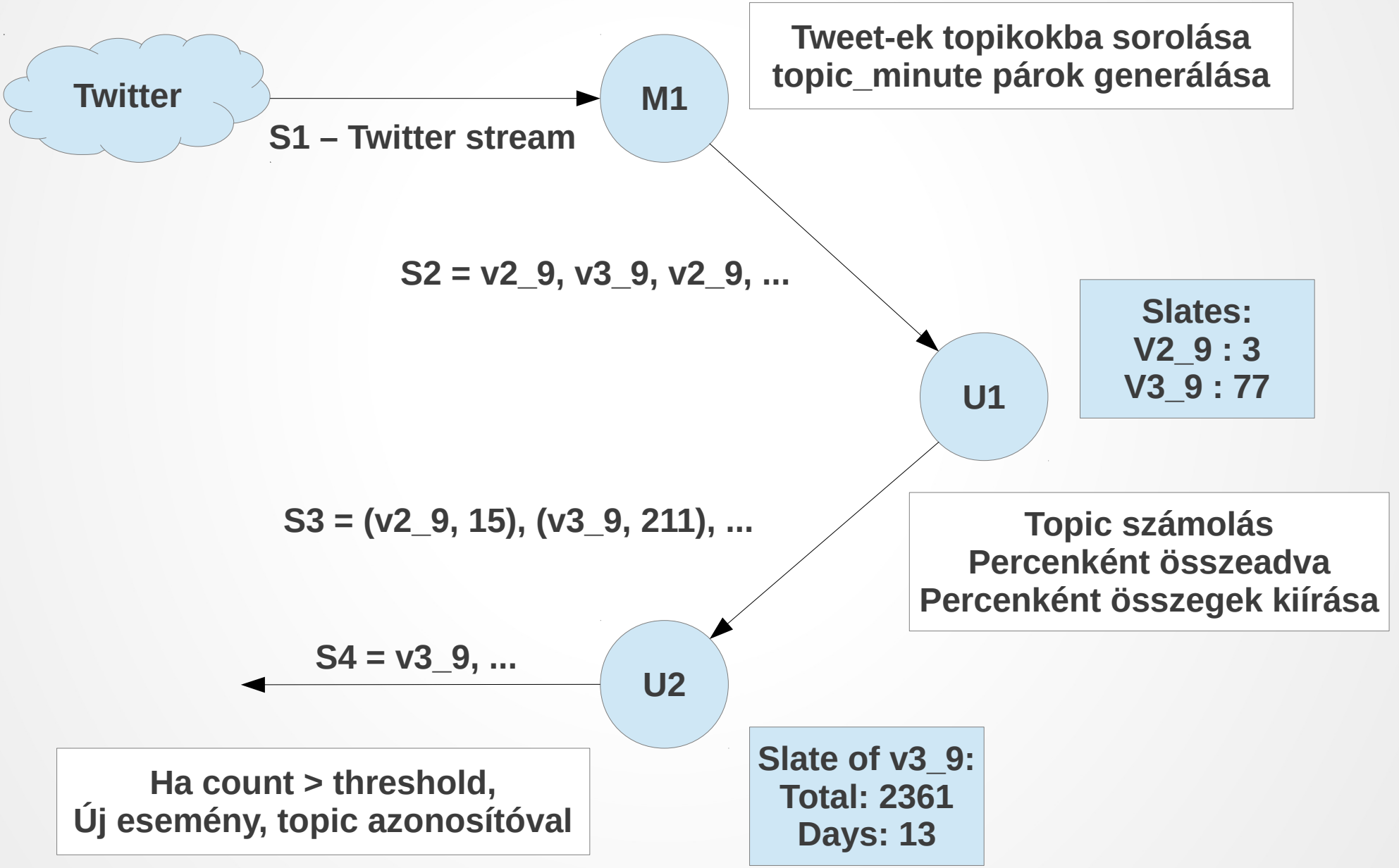
- Irányított gráf
  - Csúcsok: map vagy update függvények
  - Élek: stream-ek
  - Hurokél és kör megengedett
- Számolt értékeket külső alkalmazás a slate-ekből olvassa ki

FourSquare checkin számláló

- Map a checkin-ből helyeket generál
- U1 a helyeket megszámolja



Twitter – hot topics



Muppet 1.0

Muppet

- A MapUpdate framework implementációja
- Minden gépen elindul megadott számú program
- Mindegyik mapper vagy updater lesz
- Master – speciális mapper, a forrás alapján eseményeket generál
- Az események elosztása hash-eléssel történik

Egy esemény útja

- Master hash alapján elküldi egy adott mappernek
- A mapper queue-ba teszi, ha éppen dolgozik
- A mapper leveszi, és valahány eseményt generál
- Minden ilyen eseményt hash-el, és az alapján elküldi egy updater-nek
- Az esemény az updater queue-jába kerül

Egy esemény útja

- Az updater előkeresi az esemény kulcsához tartozó slate-et
- Végrehajtja az update-et, ami további eseményeket generálhat
- A további események hasonlóan kerülnek feldolgozásra

Események

- Az események közvetlenül, worker-ek között mennek, nem a master-en keresztül
- MapReduce-ban a master közvetít (legalábbis központi helyre kerül az adat)
- Latency minimalizálása

Slate kezelés

- Elsődleges tárolás memóriában
- Mentés key-value store-ba
  - Swap-elés lehetősége
  - A program a leállása után folytatható (leállítás, crash)
  - Gyakran a slate-ekben van hasznos adat, ezek a worker nélkül is elérhetőek így
- A Muppet Cassandra-t használ erre

Hibák kezelése

- Gép kiesése
  - Master-t értesíti az észlelő node
  - Az események átirányításra kerülnek
  - A node-on tárolt adat elveszett (kivéve az utoljára kiírt slate állapotok)
  - Az alacsony késés fontosabb, mint esetleg néhány tweet fel nem dolgozása
  - Gyors helyreállás a helyességgel szemben

Slate-ek olvasása

- A számított adat gyakran a slate-ekben van
- Kis HTTP szerver minden node-on
- Az URI tartalmazza a kulcsot és az updater-t
- Ha a slate nincs a node-on, továbbítja a kérést

Muppet 2.0

1.0

- Worker
  - Perl vezérlő
    - Események fogadása, küldése, hash-elés, stb.
  - Java feldolgozó
    - Esemény feldolgozása, újak generálása
- Több worker egy gépen
- Külön slate-ek minden worker-hez

2.0

- Java és Scala implementáció
- 1 program fut minden node-on
  - Thread pool, minden thread egy worker
  - Külön IO a key-value store-hoz (nem blokkolja a számításokat)
  - 1 slate cache / node

Esemény kezelés

- Queue minden workerre (nem node-ra)
- Esemény hash alapján 2 sorba kerülhet
- Ha valamelyikben már van azonos kulcs, oda kerül
- Ha nincs, akkor az elsőbe, kivéve, ha a második sor sokkal rövidebb
- Kisebb eséllyel telik be egy-egy sor

Eredmények

Eredmények

- 2011 elején
  - Napi 100 millió tweet, 1.5 FourSquare checkin
  - 30 millió felhasználói profil slate
  - Gépek száma 10-es nagyságrendű
  - Átfutási idő 2 sec alatt
- 2012 június
  - Több, mint 2 milliárd slate különböző alkalmazásokból

Létező alkalmazások

- TweetBeat - Tweet-ek témákba sorolása
  - Pl. Foci világbajnokság
  - Lehet országra szűrten nézni a tweet-eket
  - Stb.
- ShoppyCat – Ajándékötletek Facebook ismerősök számára

Q & A