# Sisteme de procesare a fluxului

# Context

- Odată cu popularizarea IoT, nr. dispozitivele inteligente utilizate pentru monitorizare, gestionare și service a crescut rapid.
- Sursele de date interconectate generează date noi în mod continuu, formând un număr mare, sau un flux masiv, de fluxuri de date care în cele din urmă vor copleși sistemele tradiționale de gestionare a datelor.
- Între timp, generarea de date în continuă creștere a fost însoțită de cererile crescânde pentru procesarea datelor cu latență scăzută.

# Procesarea fluxului (Stream processing)

- Dorința unei analize rapide a datelor dă naștere apariției procesării fluxului, *o nouă paradigmă de procesare în memorie care permite colectarea, analiza și vizualizarea datelor în flux cu doar latențe de secunde sau milisecunde*.
- Procesarea fluxului este o paradigmă pentru a gestiona fluxurile de date la sosire, alimentând aplicații critice pentru latență, cum ar fi detectarea fraudelor, tranzacționarea algoritmică și supravegherea sănătății

# Particularități ale procesării fluxului

- Spre deosebire de paradigma tradițională a magazinului, în primul rând, procesarea mai târziu, procesarea fluxului consumă în mod continuu datele primite pentru a oferi informații imediate
  - Datele primite sunt gestionate la sosire, rezultatele fiind actualizate progresiv în timp ce datele circulă prin sistem.
- Prezentat cu resurse limitate pentru a gestiona intrările continue, procesarea fluxului nu are acces aleatoriu la întregul flux
  - În schimb, instalează logica de procesare în ferestre bazate pe timp sau pe buffer, efectuând calcule ușoare și independente asupra datelor primite recent.
  - În acest fel, cerința strictă de latență poate fi îndeplinită prin echilibrarea adecvată a sarcinii de lucru și paralelizarea procesării pe o serie de resurse distribuite

# Procesarea distribuită sau divizarea fluxului

- Procesarea fluxului are nevoie de SLA-uri specifice privind latența de la capăt la capăt, debitul susținut al fluxului și garanția semantică de procesare pentru a face față naturii dinamice a fluxurilor de intrare și naturii partajate a infrastructurii
- Conceptul de bază din spatele motoarelor de procesare a fluxurilor distribuite este procesarea elementelor de date primite în timp real prin modelarea unui flux de date în care există mai multe etape care pot fi procesate în paralel.
- Alte tehnici includ împărțirea fluxului de date în mai multe sub-fluxuri și redirecționarea acestora într-un set de noduri în rețea

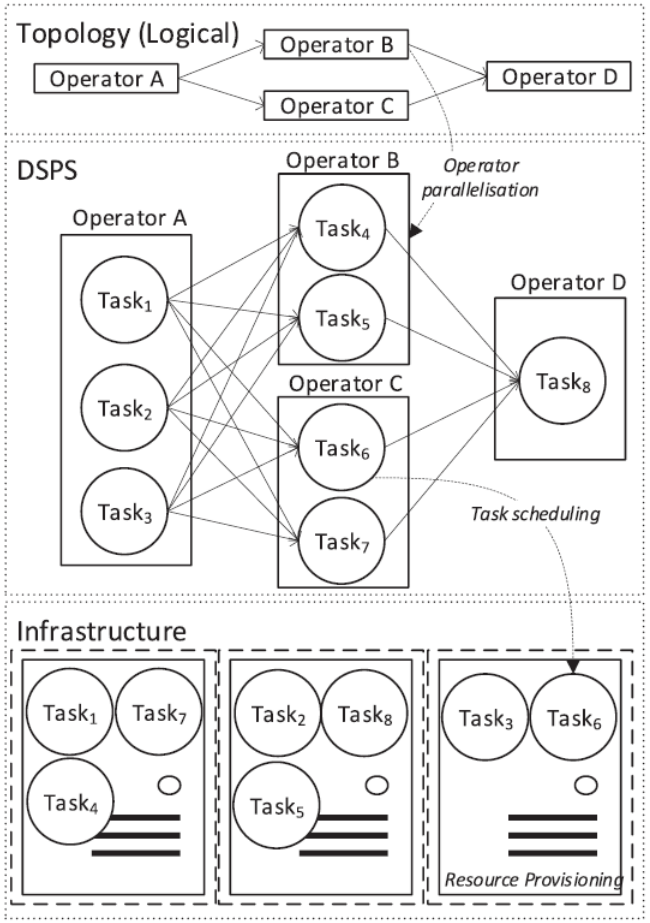
# De ce procesarea fluxului și gestionarea resurselor

- există o varietate de sisteme de procesare a fluxurilor distribuite (DSPS) care facilitează dezvoltarea aplicațiilor de streaming
- gestionarea resurselor și programarea sarcinilor nu sunt gestionate automat de middleware-ul DSPS și necesită un proces laborios pentru a se adapta la ținte specifice de implementare

# Sistem de streaming: aplicație+DSPS+infrastructură

- Din punct de vedere structural, un DSPS funcționează ca middleware al unui sistem distribuit, oferind
  - management unificat al fluxului,
  - interfețe imperative de programare a aplicațiilor (API) și
  - un set de primitive de streaming pentru a simplifica implementarea aplicației.
- DSPS-uri de ultimă generație: Apache Storm și Apache Flink
  - Oferă toleranță la erori, scalabilitate și gestionarea stării transparente pentru aplicațiile de nivel superior, reducând în același timp complexitatea coordonării resurselor distribuite.
- Un sistem de streaming tipic este astfel o structură pe trei niveluri care cuprinde:
  - aplicațiile utilizator, DSPS și infrastructura de bază.

# Structura ierarhică a unui sistem de streaming




- Streaming applications
- User-defined continuous queries

DSPS offers:

- Unified stream management
- Imperative Programming APIs
- Integrated streaming primitives
- Horizontal scalability
- State management
- Fault-tolerance

- A set of distributed resources
- An interconnected computing environment
- Deployed in clouds

*X.Liu, R. Buyya. Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions. ACM Comput. Surv. 53, 3, (May 2021),. DOI: 10.1145/3355399*

# Implementarea unui sistem de streaming

- sarcină intensivă de muncă pentru a implementa un sistem de streaming într-un mediu distribuit care satisface anumite cerințe de calitate a serviciului (QoS) cu costuri minime ale resurselor
- Trei decizii:

  (1) furnizarea de resurse - determinarea compoziției infrastructurii de procesare,

  (2) paralelizare—configurarea gradului de paralelism pentru logica de streaming și

  (3) planificarea sarcinilor — deciderea plasării sarcinilor de streaming pe resursele distribuite
- Cloud computing oferă un pool de resurse scalabil și elastic

# (1) Aprovizionarea cu resurse

- descrie activitățile de estimare, selectare și alocare a resurselor adecvate de la furnizorul de servicii pentru a constitui mediul de procesare a fluxului interconectat.
- Estimarea resurselor:
  - Estimarea tipului și cantității de resurse necesare sistemului pentru a-și îndeplini obiectivele de performanță și costuri articulate în SLA.
  - Poate fi derivat din analiza datelor istorice, precum și din predicția viitoarei sarcini de lucru
  - Precizia sa este adesea afectată de fluctuația instantanee, neașteptată a intrărilor și de variațiile de performanță ale sistemului, datorită naturii dinamice a fluxurilor de date.
- Adaptarea resurselor:
  - cererile reale de resurse pot fluctua odată cu variația volumului de muncă sau pot rămâne vagi și neclar chiar și după ce sistemul este conectat.
  - găsirea momentului potrivit în timp pentru a extinde/reduce și alegerea schemei de adaptare potrivite rămâne o provocare uriașă.
  - Rentabilitatea adaptării este afectată de un nr. factori precum modelul de facturare selectat.
  - Latența de rețea neneglijabilă trebuie luată în considerare atunci când se efectuează adaptarea sistemului într-o manieră distribuită

# (2) Paralelizare

- împarte un operator paralel în mai multe replici echivalente din punct de vedere funcțional, fiecare gestionând un subset al întregului operator pentru a accelera procesarea datelor
- Calculul paralelismului:
  - necesită profilarea precisă a volumului de lucru al fluxului și analizarea capacității de procesare a fiecărei sarcini.
  - numărul de nuclee/thread-uri dintr-un procesor limitează gradul maxim de paralelism în timpul rulării
- Ajustarea paralelismului:
  - Supra-paralelizarea și subparalelizarea pot apărea în timpul execuției ca urmare a modificării volumului de lucru sau a adaptării resurselor.
  - Provocare: monitorizați și profilați sarcinile de streaming la un nivel fin pentru a dezvălui adevăratul blocaj de performanță al aplicației.
- Echilibrare sursă de date (injectare date în grafic)/chiuve (consum doar operatorul periferic):
  - trebuie ajustate, deoarece performanțele lor sunt corelate datorită modelului de comunicare între producător și consumator din sistemul de streaming.
  - O sursă de date prea puternică poate cauza întârzieri grave în rezervoarele de date, în timp ce o sursă de date ineficientă ar înfometa operatorii următori și ar îngreuna debitul general.

# (3) Planificarea sarcinilor

- mapează dinamic resursele activităților de streaming, astfel încât fluxurile de date să fie partiționate și procesate în diferite locații simultan și independent.
- echilibrarea încărcăturii rutării fluxului se bazează pe DSPS pentru a partiționa corect fluxurile de date între sarcinile de streaming aparținând aceluiași operator
- Programarea sarcinilor pentru sistemele de procesare în flux este similară cu programarea fluxului de lucru pentru sistemele de procesare în serie
- Obiective:
  - Minimizarea comunicării între noduri
  - Atenuarea conflictului de resurse
  - Planificarea orientată spre performanță

# Apache Storm

- Motor de procesare a fluxurilor distribuite utilizat de Twitter în urma dezvoltării extinse
- Lansarea sa inițială a fost 17 septembrie 2011, iar până în septembrie 2014 a devenit open-source
- utilizat de companii precum Groupon, Yahoo!, Spotify, Verisign, Alibaba, Baidu, Yelp și multe altele
- topologia definită acționează ca o conductă distribuită de transformare a datelor.
- programele din Storm sunt concepute ca o topologie sub formă de DAG, constând din „spouts (bucuri)” și „bolts (șuruburi)”

# Spouts (bucuri)

- „Spouts” citesc datele din surse externe și le emit în topologie ca un flux de „tupluri”.
- Această structură este însoțită de o schemă care definește numele câmpurilor tuplurilor.
- Tuplurile pot conține valori primitive, cum ar fi numere întregi, lungi, scurte, octeți, șiruri de caractere, duble, floats, booleeni și matrice de octeți.
- În plus, pot fi definite serializatoare personalizate pentru a interpreta aceste date.

# Bolts (șuruburi)

- Etapele de procesare ale unui flux sunt definite în „bolts” care pot efectua manipularea datelor, filtrarea, agregarea, îmbinările și așa mai departe.
- Bolt-urile pot constitui, de asemenea, structuri de transformare mai complexe care necesită mai mulți pași (astfel, șuruburi multiple).
- Bolt-urile pot comunica cu aplicații externe, cum ar fi bazele de date și cozile Kafka

# Exemple tipice de utilizare a Storm

- Procesarea unui flux de date noi și actualizarea bazelor de date în timp real, de exemplu în sistemele de tranzacționare în care acuratețea datelor este crucială;
- Interogarea continuă și transmiterea rezultatelor către clienți în timp real, de exemplu transmiterea în browser a subiectelor trending pe Twitter,
- O paralelizare a unei interogări intensive de calcul din mers, și anume, un apel de procedură de la distanță (RPC) distribuit în care sunt testate un număr mare de seturi

# Probleme ale Storm-ului și soluții

- **Probleme**
  - Topologiile de furtună, odată create, rulează pe termen nelimitat până când sunt ucise.
    - împrăștierea ineficientă a sarcinilor aplicației între nodurile Cluster are un impact de durată asupra performanței.
  - Planificatorul implicit al Storm implementează o strategie Round Robin.
  - În scopul alocării resurselor, Storm presupune că fiecare lucrător este omogen.
    - Acest design are ca rezultat o supraalocare frecventă a resurselor și o utilizare ineficientă a comunicațiilor între sisteme
- **Soluții**
  - D-Storm din 2017 (academic)
    - Strategia sa de planificare se bazează pe un algoritm metaeuristic Greedy, care monitorizează, de asemenea, volumul volumului de lucru primit și este conștient de resurse.
  - Heron a înlocuit Storm în 2018 la Twitter (comercial)
    - noul motor de procesare a fluxurilor distribuite, Heron, care continuă abordarea modelului DAG, se concentrează pe diverse îmbunătățiri arhitecturale, cum ar fi reducerea cheltuielilor generale, testabilitatea și accesul mai ușor la datele de depanare.

# Streaming și edge computing

- procesarea fluxurilor de date continue ca o aplicație de vârf ideală, mai ales atunci când acele fluxuri de date se află la sediul utilizatorului final și au o rată de acces scăzută (de exemplu, supraveghere video)
- Promisiunea edge computing: mai puțină utilizare a lățimii de bandă în rețeaua de bază
  - De obicei, toate valorile brute ar fi transmise în flux în cloud; cu toate acestea, având în vedere creșterea datelor, acest lucru ar putea supraîncărca rețeaua de bază.
  - Acest lucru este relevant, deoarece lățimea de bandă a rețelei extinse rămâne o resursă limitată
  - Același lucru este valabil și pentru multe dintre rețelele de acces wireless actuale
  - Fluxurile de date mai mari și continue pot fi o povară pentru rețelele de backhaul.
  - Procesarea distribuită și agregarea fluxurilor de date de-a lungul căii către consumator poate ajuta la atenuarea acestui fenomen.

# Filtrarea la margine

- pentru a elimina datele irelevante
- deoarece nu toate datele sunt la fel de importante, economisirea lățimii de bandă poate fi realizată prin eliminarea datelor irelevante înainte de a fi transmise pentru procesare ulterioară.
- exemplu:
  - limitarea citirilor de temperatură într-o aplicație în care o alarmă ar trebui să fie ridicată atunci când o anumită valoare este depășită
  - citirile de temperatură sunt irelevante atâta timp cât sunt în intervalul normal și, prin urmare, nu trebuie transmise.

# Preprocesare la margine

- Datele sunt transformate de la o reprezentare la alta.
- Pe lângă economisirea lățimii de bandă, reducerea datelor la nivel local poate ajuta, de asemenea, la economisirea energiei și la reducerea nevoilor locale de stocare.
- Aruncarea datelor ar putea fi interpretată ca un caz special al unei astfel de transformări
- Alte transformări posibile: agregarea fluxurilor de date în timp, compresia datelor, modificarea datelor sau crearea de punte între formate.
- De exemplu, analiză video în timp real (o aplicație probabil ucigașă pentru edge computing),
  - redirecționarea numai a rezultatelor analizei, de exemplu, numărul de obiecte din cadru, în loc de fluxuri video întregi sau date de preprocesare pentru o aplicație de recunoaștere a feței.
- În cazul aplicațiilor de procesare a fluxului de date critice în timp, distribuirea operațiunilor în întregime la margine poate reduce în mod substanțial latențele end-to-end.