A Cost Efficient Scheduling Strategy to Guarantee Probabilistic Workflow Deadlines

Thomas Bach, Muhammad Adnan Tariq, Boris Koldehofe, Kurt Rothermel

Wednesday, 11th March 2015
Business Processes

- Workflows are the de-facto standard for business process automation
  - Usually run on dedicated hardware (e.g., workstations, servers)
- Today, businesses are going more and more pervasive
  ⇒ Workflows tightly integrate users, need for low latency

Traditional Business Processes

User Centric Business Processes
Business Processes

40.8 network failures per day with end-user impact. Median packet loss of 59,000 packets per failure.

4G LTE: 16% Network failures out of 100 requests
3G: 19% network failures out of 100 requests

2013 US Study by J.D. Power

Microsoft Datacenter

User Centric Business Processes

A 2 second slowdown changed queries/user by -1.8% and revenue/user by -4.3%.

Eric Schurman (Bing), Velocity 2009, San Jose

Response Time Measurement of 4532 traditional Web-Services – Response Time Distribution has a long tail and is heavily dependent on user location.

Zheng et. Al.

Every 100ms delay costs Amazon 1% of sales

Make data useful G. Linden - gduchamp.com

$197.4 million Q1 2014, Forbes

A 2 second slowdown changed queries/user by -1.8% and revenue/user by -4.3%.

User Centric Business Processes

Scheduling of “the right” services for the different activities
Contributions

We address the problem of scheduling services to guarantee probabilistic workflow deadlines at minimum costs

1. Efficiently schedule services for one activity
2. Derive activity level deadlines form final deadline
3. Perform scheduling in the presence of dynamics
1. Execution engine executes activities sequentially
   - Idempotent
     • Arbitrarily often (e.g., checking ticket availability)
   - Non idempotent
     • At most once (e.g., buying a ticket)
     • Compensation required
2. ESB used to locate and invoke services
   - Service characterized by response time distribution
   - Invocation and compensation cost
     ⇒ Invocation (e.g. communication, computation)
     ⇒ Compensation (e.g. cancelation fee, communication overhead)
Contributions

We address the problem of scheduling services to guarantee probabilistic workflow deadlines at minimum costs

1. Efficiently schedule services for one activity
Parallel vs. Sequential Service Invocation

Parallel → overprovisioning
- Overprovisioning
- High cost

Sequential → too late
- Deadline violated

Related work does not take type of activity into account

Activity-level Approach

Search space between parallel and sequential

Available Services

- Only invoke Service when previous not responds
- Select services from available services
- Find a schedule \( \tau = \{\tau_{s_1}, \tau_{s_2}, ..., \tau_{s_n}\} \) of services s.t. minimal number services is executed
- \( \Rightarrow \) NP hard problem, approach: Simulated Annealing, tailored neighborhood function
Activity-level Neighborhood Functions

- Neighborhood function consists of two components
  - Select services to change
  - Change start points of services

1560 experiments per neighborhood function, real service response times
Activity-level Scheduling

- Services reduced by 70%
- ~200,000 times faster
- 0.39% worse than Brute Force
Contributions

We address the problem of scheduling services to guarantee probabilistic workflow deadlines at minimum costs

1. Efficiently schedule services for one activity
2. Derive activity level deadlines form final deadline
Workflow-level Scheduling

Different cost for different activity types
- Give more time to non-idempotent activities

Simulated Annealing solution:
1. Start with equally distributed deadlines
2. Change activity deadline
3. Reschedule services for each activity
Evaluation Workflow-level scheduling

20 Activities, 50% idempotent, compensation cost 100x execution, 3000 experiments

Expected costs reduced by 70%
Contributions

We address the problem of scheduling services to guarantee probabilistic workflow deadlines at minimum costs.

1. Efficiently schedule services for one activity
2. Derive activity level deadlines from final deadline
3. Perform scheduling in the presence of dynamics
Why Dynamic Scheduling is Needed

Computational Overhead

Balanced according to costs

Reached with high probability

Finish earlier

Idle time building up

Service characteristics change

idempotent

Non idempotent

idempotent

Check Ticket Availability

Buy Ticket

Check Train Timing

\(d_{a_1}\)

\(d_{a_2}\)

\(d_{final}\)
Dynamic Workflow-level Scheduling

- Sliding scheduling window to take real execution time into account

  While not all activities executed:
  1. Divide remaining time between all activities
  2. Only schedule the next $k$ activities with activity-level-scheduling
Dynamic Workflow-level scheduling

100 Activities, 50% idempotent, Compensation cost 100x execution, 900 experiments

Cost reduced by 85%
Conclusion and Future Work

Conclusion
We addressed the problem of scheduling services to guarantee probabilistic workflow deadlines at minimum costs:

1. 70% less services uses per activity
2. Decreased expected number of services on Workflow level by 70%
3. Dynamic scheduling decreased real execution cost by 85%

Future Work
- Take branching in workflows into account
Thank you for your attention.

A Cost Efficient Scheduling Strategy to Guarantee Probabilistic Workflow Deadlines

Thomas Bach, Muhammad Adnan Tariq, Boris Koldehofe, Kurt Rothermel

Institute for Parallel and Distributed Systems,
University of Stuttgart, Germany
thomas.bach@ipvs.uni-stuttgart.de