Mean field and fluid approaches to Markov chain analysis

Jeremy Bradley
Imperial College London

Invited talk, QAPL 2012

31 April 2012
Mean field/fluid analysis

- Addresses the state-space explosion problem for discrete-state Markov models of computer and communication systems

- Derives tractable systems of differential equations approximating mean number of components in each local state, for example:
  - Fluid analysis of process algebra models[1]
  - Mean-field analysis of systems of interacting objects[2,3]

- Can develop these techniques to capture key performance measures of interest from large CTMCs, e.g. passage-time measures, reward-based measures


Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

Moment ODEs
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

- Moment ODEs
- Passage times

Optimal SLA satisfaction

Energy consumption

SLA H
SLA L

Time, t
Consumption Faster
Slower

Nslow
Nfast

Both SLAs
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

Moment ODEs

Passage times

SLA: \( \leq 6.5 \text{s w.p. } \geq 90\% \)

![Graph showing SLA and passage times]

Time, \( t \)

Probability

Faster

Slower

Energy consumption

SLA H

SLA L

Both SLAs

Energy consumption

N slow

N fast
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation
(e.g. Grouped PEPA, Cts SPNs)

Moment ODEs

- Passage times
- Rewards
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

- Moment ODEs
- Passage times
- Rewards

Optimal SLA satisfaction

Energy consumption

Both SLAs

N_{slow} N_{fast}
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation
(e.g. Grouped PEPA, Cts SPNs)

- Moment ODEs
- Passage times
- Rewards
- Optimal SLA satisfaction
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

- Moment ODEs
- Passage times
- Rewards
- Optimal SLA satisfaction

Diagram:
- Energy consumption
- SLA H
- SLA L
- Both SLAs
- $N_{\text{slow}}$, $N_{\text{fast}}$
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

Moment ODEs

Passage times

Rewards

Optimal SLA satisfaction
Overview

Massively-parallel Markov models

- Grouped PEPA (stochastic process algebra)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Structured CTMC representation (e.g. Grouped PEPA, Cts SPNs)

Moment ODEs

Passage times

Rewards

Optimal SLA satisfaction

SLA: \( \leq 6.5 \text{s w.p.} \geq 90\% \)

Time, \( t \)

Probability

Energy consumption

SLA H

SLA L

Both SLAs
A simple agent
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated
A simple agent – replicated

Fluid/mean field analysis works best when you have many replicated parallel agents or groups of replicated parallel agents. Agent groups can synchronise.
Abstract client/server model

\[
\text{Population CTMC state space: } (N_C(t), N_C^W(t), N_C^P(t), N_S(t), N_S^P(t), N_S^F(t)) \in \mathbb{Z}^6_+
\]

State space grows exponentially as component types are added.

Explicit-state analysis techniques do not scale.
Abstract client/server model

- Client
  - \( Client_{\text{wait}} \)
  - req

- Server
  - \( Server_{\text{proc}} \)
  - req

Population CTMC state space:
\[
\left( N_C(t), N_C(W(t)), N_C(P(t)), N_S(t), N_S(P(t)), N_S(F(t)) \right) \in \mathbb{Z}_6^+.
\]

- State space grows exponentially as component types are added.
- Explicit-state analysis techniques do not scale.
Abstract client/server model

Population CTMC state space:

$$\left( N_C(t), N_C(W(t)), N_C(P(t)), N_S(t), N_S(P(t)), N_S(F(t)) \right) \in \mathbb{Z}^{6}_+$$

State space grows exponentially as component types are added

Explicit-state analysis techniques do not scale.
Abstract client/server model

- **Client**
  - Client\_wait
  - Client\_proc

- **Server**
  - Server\_proc

- **proc**
  - req
  - res

- **Population CTMC state space:**
  \[(N_{C}(t), N_{C}(W)(t), N_{C}(P)(t), N_{S}(t), N_{S}(P)(t), N_{S}(F)(t)) \in \mathbb{Z}_{+}^{6}\]

- State space grows exponentially as component types are added

- Explicit-state analysis techniques do not scale
Abstract client/server model

Client

\[ \text{Client}_{\text{wait}} \]

\[ \text{Client}_{\text{proc}} \]

Server$_{\text{fail}}$

\[ \text{Server} \]

\[ \text{Server}_{\text{proc}} \]

\[ \text{req} \quad \text{res} \]

\[ \text{req} \quad \text{reset} \quad \text{fail} \]

\[ \text{proc} \]

\[ \text{proc} \]

\[ \text{State space grows exponentially as component types are added} \]

\[ \text{Explicit-state analysis techniques do not scale} \]
Abstract client/server model

Population CTMC state space:

\[
\left( N_C(t), N_C(W(t)), N_C(P(t)), N_S(t), N_S(P(t)), N_S(F(t)) \right) \in \mathbb{Z}^{6+}
\]

State space grows exponentially as component types are added.

Explicit-state analysis techniques do not scale.
Abstract client/server model

Markovian dynamics depend on the aggregate rate of \( \text{req} \)-synchronisations, for example:

\[
\alpha \min(N_C(t), N_S(t))
\]

bounded capacity, resource constrained
e.g. PEPA, Petri nets, queueing nets

\[
(\alpha/N_S)N_C(t)N_S(t)
\]

mass-action, e.g. peer-to-peer nets
Abstract client/server model

Population CTMC state space:

\[(N_C(t), N_{CW}(t), N_{CP}(t), N_S(t), N_{SP}(t), N_{SF}(t)) \in \mathbb{Z}_+^6\]
Abstract client/server model

- Population CTMC state space:
  \[(N_C(t), N_{CW}(t), N_{CP}(t), N_S(t), N_{SP}(t), N_{SF}(t)) \in \mathbb{Z}_+^6\]

- State space grows exponentially as component types are added
Abstract client/server model

Population CTMC state space:

\[(N_C(t), N_{CW}(t), N_{CP}(t), N_S(t), N_{SP}(t), N_{SF}(t)) \in \mathbb{Z}^6_+\]

- State space grows exponentially as component types are added
- Explicit-state analysis techniques do not scale
Approximate moment ODEs

Simulation is also very costly for large populations.
Approximate moment ODEs

- Simulation is also very costly for large populations.
Approximate moment ODEs

▶ Simulation is also very costly for large populations.
Approximate moment ODEs

- Can derive coupled ODEs for component-count moments:

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = \cdots
\]
Approximate moment ODEs

Can derive coupled ODEs for component-count moments:

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = \cdots \quad \frac{d}{dt} \text{Var}[N_C(t)] = \cdots
\]
Can derive coupled ODEs for component-count moments:

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = \cdots \quad \frac{d}{dt} \text{Var}[N_C(t)] = \cdots \quad \frac{d}{dt} \mathbb{E}[N_S(t)N_C^2(t)] = \cdots
\]
Approximate moment ODEs

Can derive coupled ODEs for component-count moments:[1,4–6]

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = \cdots \quad \frac{d}{dt} \text{Var}[N_C(t)] = \cdots \quad \frac{d}{dt} \mathbb{E}[N_S(t)N_C^2(t)] = \cdots
\]


Approximate moment ODEs

\[
\begin{align*}
\text{Client} & \xrightarrow{\text{req}, \alpha} \text{Client}_{\text{wait}} \\
\text{Client}_{\text{wait}} & \xrightarrow{\text{res}} \text{Client}_{\text{proc}} \\
\text{Server}_{\text{fail}} & \xrightarrow{\text{reset}} \text{Server} \\
\text{Server} & \xrightarrow{\text{req}, \alpha} \text{Server}_{\text{proc}} \\
\end{align*}
\]

\[\begin{align*}
\text{proc}, \beta \\
N_C \\
N_S
\end{align*}\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \mathbb{E}[\min(N_C(t), N_S(t))] + \beta \mathbb{E}[N_{Cp}(t)]
\]

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C(t)N_S(t)] + \beta \mathbb{E}[N_{Cp}(t)]
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \mathbb{E}[\min(N_C(t), N_S(t))] + \beta \mathbb{E}[N_{C_p}(t)] \\
\approx -\alpha \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) + \beta \mathbb{E}[N_{C_p}(t)] \\
\frac{d}{dt} \mathbb{E}[N_C(t)] = - (\alpha/N_S) \mathbb{E}[N_C(t)N_S(t)] + \beta \mathbb{E}[N_{C_p}(t)]
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \mathbb{E}[\min(N_C(t), N_S(t))] + \beta \mathbb{E}[N_{C_p}(t)]
\]

\[
\dot{v}_C(t) = -\alpha \min(v_C(t), v_S(t)) + \beta v_{C_p}(t)
\]

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C(t)N_S(t)] + \beta \mathbb{E}[N_{C_p}(t)]
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \mathbb{E}[\min(N_C(t), N_S(t))] + \beta \mathbb{E}[N_{Cp}(t)]
\]

\[
\dot{v}_C(t) = -\alpha \min(v_C(t), v_S(t)) + \beta v_{Cp}(t)
\]

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C(t)N_S(t)] + \beta \mathbb{E}[N_{Cp}(t)]
\]

\[
\approx -\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C(t)]\mathbb{E}[N_S(t)] + \beta \mathbb{E}[N_{Cp}(t)]
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \mathbb{E}\left[\min(N_C(t), N_S(t))\right] + \beta \mathbb{E}[N_{C_p}(t)]
\]
\[
\dot{v}_C(t) = -\alpha \min(v_C(t), v_S(t)) + \beta v_{C_p}(t)
\]
\[
\frac{d}{dt} \mathbb{E}[N_C(t)] = -\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C(t)N_S(t)] + \beta \mathbb{E}[N_{C_p}(t)]
\]
\[
\dot{v}_C(t) = -\left(\frac{\alpha}{N_S}\right) v_C(t)v_S(t) + \beta v_{C_p}(t)
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots
\]
Approximate moment ODEs

\[ \frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots \]

\[ \approx -2\alpha \min(\mathbb{E}[N_C^2(t)], \mathbb{E}[N_C(t)N_S(t)]) + \ldots \]

\[ \frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots \]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\dot{v}_{C^2}(t) = -2\alpha \min(v_{C^2}(t), v_{C \cdot S}(t)) + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\dot{\nu}_{C^2}(t) = -2\alpha \min(\nu_{C^2}(t), \nu_{C \cdot S}(t)) + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots
\]

\[
\approx - (\alpha/N_S) \mathbb{E}[N_C^2(t)]\mathbb{E}[N_S(t)] + \ldots
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\dot{v}_{C^2}(t) = -2\alpha \min(v_{C^2}(t), v_{C \cdot S}(t)) + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\left(\frac{\alpha}{N_S}\right) \mathbb{E}[N_C^2(t) N_S(t)] + \ldots
\]

\[
\dot{v}_{C^2}(t) = -\left(\frac{\alpha}{N_S}\right) v_{C^2}(t) v_S(t) + \ldots
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\dot{\nu}_{C^2}(t) = -2\alpha \min(\nu_{C^2}(t), \nu_{C \cdot S}(t)) + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots
\]

\[
\approx -\left(\frac{\alpha}{N_S}\right)\mathbb{E}[N_C(t)]\mathbb{E}[N_C \cdot S(t)] + \ldots
\]
Approximate moment ODEs

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2\alpha \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + \ldots
\]

\[
\dot{v}_{C_2}(t) = -2\alpha \min(v_{C_2}(t), v_{C \cdot S}(t)) + \ldots
\]

\[
\frac{d}{dt} \mathbb{E}[N_C^2(t)] = -2(\alpha/N_S) \mathbb{E}[N_C^2(t)N_S(t)] + \ldots
\]

\[
\dot{v}_{C_2}(t) = -(\alpha/N_S) v_C(t)v_{C \cdot S}(t) + \ldots
\]
Switch points

For bounded capacity Markovian dynamics (e.g. in Petri Nets):

$$\frac{d}{dt} \mathbb{E}[N_C(t)] = -\alpha \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) + \beta \mathbb{E}[N_{C_p}(t)]$$

Switch points capture the approximation:

$$\mathbb{E}[\min(N_C(t), N_S(t))] \approx \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)])$$
Example — mean

![Graph showing the component count over time for different server states. The graph compares the numbers of servers (Server), server processes (Server$_{proc}$), and failed servers (Server$_{fail}$) over time.]
Example — standard deviation

![Graph showing component count standard deviation over time for different servers.](image)
Example — skewness

![Graph showing component count skewness over time for Server, Server\(_{proc}\), and Server\(_{fail}\).]
Scalable passage-time analysis
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  “file should be transferred within 2 seconds, 95% of the time”


Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:
  
  “connection should be established within 0.25 seconds, 99% of the time”
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:
  
  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  “connection should be established within 0.25 seconds, 99% of the time”

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:
  
  “connection should be established within 0.25 seconds, 99% of the time”

- We consider two classes of passage-time query:
  
  - **Individual passage times**: track the time taken for an individual to complete a task

  - **Global passage times**: track the time taken for all of a large number of individuals to complete a task
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  “connection should be established within 0.25 seconds, 99% of the time”

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task
    - Direct approximation to the entire CDF
  - Global passage times: track the time taken for all of a large number of individuals to complete a task
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:
  
  “connection should be established within 0.25 seconds, 99% of the time”

- We consider two classes of passage-time query:
  
  - Individual passage times: track the time taken for an individual to complete a task
    - Direct approximation to the entire CDF
  
  - Global passage times: track the time taken for all of a large number of individuals to complete a task
    - Moment-derived bounds on CDF
How long does it take a single client to make a request, receive a response and process it?

How long does it take a single client to make a request, receive a response and process it?
Individual passage times

How long does it take a single client to make a request, receive a response and process it?
Individual passage times

\[ T := \inf \{ t \geq 0 : C(t) = \text{Client}' \}, \text{ given that } C(0) = \text{Client} \]
Individual passage times

\[ T := \inf \{ t \geq 0 : C(t) = \text{Client}' \}, \text{ given that } C(0) = \text{Client} \]

\[ \mathbb{P} \{ T \leq t \} = \mathbb{P} \{ C(t) \in \{ \text{Client}', \text{Client}'_{\text{wait}}, \text{Client}'_{\text{proc}} \} \} \]
Individual passage times

\[ T := \inf\{ t \geq 0 : C(t) = \text{Client}' \} \]

\[
P\{ T \leq t \} = P\{ C(t) \in \{ \text{Client}', \text{Client}'_{\text{wait}}, \text{Client}'_{\text{proc}} \} \}
\]

\[
= E[1 \{ C(t) = \text{Client}' \}] + E[1 \{ C(t) = \text{Client}'_{\text{wait}} \}] + E[1 \{ C(t) = \text{Client}'_{\text{proc}} \}]
\]
Individual passage times

\[ T := \inf \{ t \geq 0 : C(t) = \text{Client}' \}, \text{ given that } C(0) = \text{Client} \]

\[ \mathbb{P}\{ T \leq t \} = \mathbb{P}\{ C(t) \in \{ \text{Client}', \text{Client}'_{\text{wait}}, \text{Client}'_{\text{proc}} \} \} \]

\[ = \mathbb{E}[1\{ C(t) = \text{Client}' \}] + \mathbb{E}[1\{ C(t) = \text{Client}'_{\text{wait}} \}] + \mathbb{E}[1\{ C(t) = \text{Client}'_{\text{proc}} \}] \]

\[ = \mathbb{E}[N_{\text{Client}'}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{wait}}}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{proc}}}(t)] \]
Individual passage times

\[ T := \inf\{ t \geq 0 : C(t) = \text{Client}' \} \], given that \( C(0) = \text{Client} \)

\[
\mathbb{P}\{ T \leq t \} = \mathbb{P}\{ C(t) \in \{ \text{Client}', \text{Client}'_{\text{wait}}, \text{Client}'_{\text{proc}} \} \}
\]

\[
= \mathbb{E}[\mathbf{1}_{\{ C(t) = \text{Client}' \}}] + \mathbb{E}[\mathbf{1}_{\{ C(t) = \text{Client}'_{\text{wait}} \}}] + \mathbb{E}[\mathbf{1}_{\{ C(t) = \text{Client}'_{\text{proc}} \}}]
\]

\[
= \mathbb{E}[N_{\text{Client}'}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{wait}}}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{proc}}}(t)]
\]

\[
\mathbb{P}\{ T \leq t \} \approx v_{\text{Client}'}(t) + v_{\text{Client}'_{\text{wait}}}(t) + v_{\text{Client}'_{\text{proc}}}(t)
\]
Example — individual passage time

\[
\begin{align*}
N_C &= 10, \ N_S = 6 \\
N_C &= 20, \ N_S = 12 \\
N_C &= 50, \ N_S = 30 \\
N_C &= 100, \ N_S = 60 \\
N_C &= 200, \ N_S = 120
\end{align*}
\]

ODE approximation

Time, \( t \)

Probability
Global passage times

How long does it take for half of the clients to make a request, receive a response and process it?


How long does it take for **half of the clients** to make a request, receive a response and process it?
Global passage times

\[ T := \inf\{ t \geq 0 : N_{C'}(t) + N_{C_w}(t) + N_{C_p}(t) \geq N_C / 2 \} \]
Global passage times

\[ T := \inf\{ t \geq 0 : N_C'(t) + N_{C_w}(t) + N_{C_p}(t) \geq N_C / 2 \} \]

Point-mass approximation:

\[ T \approx \inf\{ t \geq 0 : v_{C'}(t) + v_{C_w}(t) + v_{C_p}(t) \geq N_C / 2 \} \]
Global passage times

- $N_C = 10, N_S = 6$
- $N_C = 20, N_S = 12$
- $N_C = 50, N_S = 30$
- $N_C = 300, N_S = 180$
- $N_C = 500, N_S = 300$
Global passage times

Point-mass approximation:

\[ T \approx \inf\{ t \geq 0 : v_{C'}(t) + v_{C'_w}(t) + v_{C'_p}(t) \geq N_C / 2 \} \]

- Approximation is very coarse
- Cannot be applied directly to the same question for all clients
Global passage times — moment bounds

▶ Moment approximations to component counts contain information about the distribution of $T^{[7]}$

Global passage times — moment bounds

- Moment approximations to component counts contain information about the distribution of $T$

- Reduced moment problem — find maximum and minimum bounding distributions subject to limited moment information$^{[10]}$

Global passage bounds — first moments

Half of the clients:

Three quarters of the clients:

All of the clients:

Global passage bounds — higher moments

Time, $t$

Probability

1st order

2nd order

4th order

CDF

Scalable analysis of accumulated reward measures
Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

\[
\text{total energy}(t) = r_S \int_{t_0}^{t} N_S(u) \, du + r_{Sp} \int_{t_0}^{t} N_{Sp}(u) \, du
\]

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

\[ \text{total energy}(t) = r_S \int_0^t N_S(u) \, du + r_{Sp} \int_0^t N_{Sp}(u) \, du \]

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

\[ \text{Server} \]
\[ \text{Server}_{\text{proc}} \]

\[ \text{Server}_{\text{proc}} \ r_{Sp} \]
\[ \text{Server} \ r_{S} \]
\[ \text{Server}_{\text{fail}} \ 0 \]

\[ \text{Time, } t \]

Accumulated reward measures

- Cost, energy, heat, …
- Constant rate

\[
\text{total energy}(t) = \int_{0}^{t} r_{S}(u) \, du + \int_{0}^{t} r_{Sp}(u) \, du
\]

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

---

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

![Diagram showing accumulated reward measures](image)

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

Accumulated reward measures

- Cost, energy, heat, …
- Constant rate

\[ \text{total energy}(t) = r_S \int_0^t N_S(u) \, du + r_{Sp} \int_0^t N_{Sp}(u) \, du \]

Moment approximations of accumulated rewards

\[ \int_0^t S_p(u) \, du \]

- Simulation also very costly for rewards

Moment approximations of accumulated rewards

![Graph showing accumulated quantity over time]

- Simulation also very costly for rewards

Moment approximations of accumulated rewards

Simulation also very costly for rewards

Moment approximations of accumulated rewards

Simulation also very costly for rewards

Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_{S_p}(u) \, du \right] = \cdots
\]

Moment approximations of accumulated rewards

Simulation also very costly for rewards

Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t S_p(u) \, du \int_0^t S(u) \, du \right] = \cdots
\]

Moment approximations of accumulated rewards

![Graph showing accumulated quantity over time](image)

- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t S_p(u) \, du \int_0^t N_S(u) \, du \right] = \cdots
\]

Moment approximations of accumulated rewards

- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_{S_p}(u) \, du \int_0^t N_S(u) \, du \right] = \cdots
\]

Moment approximations of accumulated rewards

Simulation also very costly for rewards

Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_{S_p}(u) \, du \int_0^t N_S(u) \, du \right] = \cdots
\]

Moment approximations of accumulated rewards

First-order moments

\[ \frac{d}{dt} E \left[ \int_0^t N_S(u) \, du \right] = \]

Moment approximations of accumulated rewards

**First-order moments**

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

Moment approximations of accumulated rewards

**First-order moments**

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

**Second-order moments**

\[
\frac{d}{dt} \mathbb{E}\left[ (\int_0^t N_S(u) \, du)^2 \right] = 2 \mathbb{E}\left[ N_S(t) \int_0^t N_S(u) \, du \right] + \cdots
\]

---

Moment approximations of accumulated rewards

First-order moments

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

Second-order moments

\[
\frac{d}{dt} \mathbb{E} \left[ \left( \int_0^t N_S(u) \, du \right)^2 \right] = \cdot \cdot \cdot
\]

Moment approximations of accumulated rewards

First-order moments

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

Second-order moments

\[
\frac{d}{dt} \mathbb{E} \left[ \left( \int_0^t N_S(u) \, du \right)^2 \right] = 2\mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right]
\]

Moment approximations of accumulated rewards

First-order moments
\[
\frac{d}{dt} \mathbb{E}\left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

Second-order moments
\[
\frac{d}{dt} \mathbb{E}\left[ \left( \int_0^t N_S(u) \, du \right)^2 \right] = 2 \mathbb{E}\left[ N_S(t) \int_0^t N_S(u) \, du \right]
\]

Combined moments
\[
\frac{d}{dt} \mathbb{E}\left[ N_S(t) \int_0^t N_S(u) \, du \right] = \ldots
\]

Moment approximations of accumulated rewards

First-order moments
\[ \frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)] \]

Second-order moments
\[ \frac{d}{dt} \mathbb{E} \left[ \left( \int_0^t N_S(u) \, du \right)^2 \right] = 2 \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] \]

Combined moments
\[ \frac{d}{dt} \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] = \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] + \cdots + \mathbb{E}[N_S^2(t)] \]

Moment approximations of accumulated rewards

First-order moments

\[ \frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E} [N_S(t)] \]

Second-order moments

\[ \frac{d}{dt} \mathbb{E} \left[ \left( \int_0^t N_S(u) \, du \right)^2 \right] = 2 \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] \]

Combined moments

\[ \frac{d}{dt} \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] = \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] + \cdots + \mathbb{E} [N_S^2(t)] \]

Moment approximations of accumulated rewards

First-order moments
\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_S(u) \, du \right] = \mathbb{E}[N_S(t)]
\]

Second-order moments
\[
\frac{d}{dt} \mathbb{E} \left[ (\int_0^t N_S(u) \, du)^2 \right] = 2 \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right]
\]

Combined moments
\[
\frac{d}{dt} \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] = \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] + \cdots + \mathbb{E}[N_S^2(t)]
\]

Trade-off between energy and performance
Trade-off between energy and performance

Client

\[ \text{Client} \]

[req]

\[ \text{Client}_{\text{wait}} \]

[res]

\[ \text{Client}_{\text{proc}} \]

\[ \text{Server}_{\text{fail}} \]

[reset]

[fail]

\[ \text{Server}_{\text{proc}} \]

\[ N_{C} \]

\[ N_{S} \]

\[ \mathbb{E}[\text{total energy}(t)] \]

\[ 0 2 4 6 8 10 \]

\[ 0.6 0.8 1 \]

\[ \text{Time, } t \]

\[ \text{Probability} \]

\[ \text{Energy} \]

\[ 0 50 100 150 \]

\[ \text{Time, } t \]

Client serviced before \( t \)
Trade-off between energy and performance

![Diagram of client and server processes](image)

- **Client**
  - Client
  - Client\_wait
  - Client\_proc

- **Server**
  - Server
  - Server\_fail
  - Server\_proc

- **Probabilities**
  - Probability of Client serviced before time $t$
  - SLA: 7 seconds, $\geq 0.99$

- **Energy**
  - Total energy over time $t$

- **Quantities**
  - $N_C$ for Client
  - $N_S$ for Server
Trade-off between energy and performance

Client

\[ \text{Client} \rightarrow \text{Client}_{\text{wait}} \rightarrow \text{Client}_{\text{proc}} \]

\[ \text{proc} \]

\[ N_C \]

Server

\[ \text{Server} \rightarrow \text{Server}_{\text{proc}} \]

\[ \text{Server}_{\text{fail}} \rightarrow \text{Server}_{\text{sleep}} \]

\[ \text{fail} \rightarrow \text{reset} \rightarrow \text{sleep/wakeup} \]

\[ N_S \]

\[ E[\text{total energy}(t)] \]

\[ \mathbb{E}[\text{total energy}(t)] \]

\[ 0 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 10 \]

\[ 0 \rightarrow 50 \rightarrow 100 \rightarrow 150 \]

\[ \text{Time, } t \rightarrow \text{Time, } t \]

\[ \text{Probability} \rightarrow \text{Energy} \]

Client serviced before \( t \)

SLA

\[ 7 \text{s} \geq 0.99 \]

0.6

0.8

1

0

2

4

6

8

10

0

50

100

150

0

2

4

6
Trade-off between energy and performance

Client

\[ \text{Client}_{\text{wait}} \]
\[ \text{Client}_{\text{proc}} \]

Server

\[ \text{Server}_{\text{fail}} \]
\[ \text{Server}_{\text{sleep}} \]

\[ \text{prob} \]
\[ \text{E}[\text{total energy}(t)] \]

\[ \text{SLA \quad 7s} \geq 0.99 \]

\[ N_C \]
\[ N_S \]
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Individual passage-time SLA: clients must finish in at most 7 seconds of the time
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA:
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA: clients must finish in at most 7s $\geq 99\%$ of the time
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA: clients must finish in at most 7s $\geq 99\%$ of the time
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA: clients must finish in at most 7s $\geq 99\%$ of the time
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA: clients must finish in at most 7s \( \geq 99.5\% \) of the time
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations ($N_S$, sleep rate)

Minimise energy consumption while satisfying SLAs

<table>
<thead>
<tr>
<th>Energy consumption</th>
<th>NS</th>
<th>rsleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1,000</td>
<td>40</td>
<td>0.4</td>
</tr>
<tr>
<td>1,200</td>
<td>80</td>
<td>0.6</td>
</tr>
<tr>
<td>800</td>
<td>60</td>
<td>0.8</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>921.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>841.48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Individual passage-time SLA: clients must finish in at most 7s ≥ 99.5% of the time
Non-Markovian models
Non-Markovian models

- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions
Non-Markovian models

- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions
- Phase-type approximation is one approach, but can lead to significant increase in a component’s local state-space size
  - A 100-phase Erlang approximation to a deterministic distribution of duration 1 has a probability of about 32% of lying outside of [0.9, 1.1]
Non-Markovian models

- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions

- Phase-type approximation is one approach, but can lead to significant increase in a component’s local state-space size
  - A 100-phase Erlang approximation to a deterministic distribution of duration 1 has a probability of about 32% of lying outside of [0.9, 1.1]

- In the case of deterministic distributions, mean-field approach can be generalised using delay differential equations
Software update model with deterministic timeouts

Software update model with deterministic timeouts

\[ \dot{E}[N_c(t)] = -\rho E[N_c(t)] - \frac{\beta}{N} E[N_c(t)N_a(t)] + \lambda E[N_e(t)] \]

\[ -\mathbb{E}\left[ \mathbf{1}_{\{t \geq \gamma\}} \lambda N_e(t - \gamma) \exp \left( -\int_{t-\gamma}^{t} \frac{\beta N_a(s)}{N} \, ds \right) \exp(-\rho \gamma) \right] \]

Rate of determ.
clocks starting at \( t-\gamma \)

Software update model with deterministic timeouts

\[ \dot{E}[N_c(t)] = -\rho E[N_c(t)] - \frac{\beta}{N} E[N_c(t)N_a(t)] + \lambda E[N_e(t)] \]

\[ - E \left[ 1_{\{t \geq \gamma\}} \lambda N_e(t-\gamma) \exp \left( - \int_{t-\gamma}^{t} \frac{\beta N_a(s)}{N} ds \right) \exp(-\rho \gamma) \right] \]

Prob. that timeout occurs before node updated or went off

Software update model with deterministic timeouts

\[ \dot{E}[N_c(t)] \approx -\rho E[N_c(t)] - \frac{\beta}{N} E[N_c(t)] E[N_a(t)] + \lambda E[N_e(t)] \\
- 1_{\{t \geq \gamma\}} \lambda E[N_e(t - \gamma)] \exp \left( - \int_{t-\gamma}^{t} \frac{\beta E[N_a(s)]}{N} \, ds \right) \exp(-\rho \gamma) \]

Software update model with deterministic timeouts

\[
\dot{v}_c(t) = -\rho v_c(t) - \frac{\beta}{N} v_c(t) v_a(t) + \lambda v_e(t) \\
- 1_{t \geq \gamma} \lambda v_e(t - \gamma) \exp \left( - \int_{t - \gamma}^{t} \frac{\beta v_a(s)}{N} \, ds \right) \exp(-\rho \gamma)
\]

Software update model with deterministic timeouts

![Graph 1](image1.png)

- Nodes in state c
- Nodes in state d
- Nodes in state e

![Graph 2](image2.png)

- Nodes in state a
- Nodes in state b
Summary

Fluid analysis provides a scalable analysis framework for massively-parallel performance models, that is able to capture:

- Arbitrary moments of component counts
- Passage-time measures
- Accumulated reward measures
- Certain forms of non-Markovian timing

with implementation in the freely-available GPA tool\(^1\)

\(^1\) [http://code.google.com/p/gpanalyser/](http://code.google.com/p/gpanalyser/)
Thank you!\(^2\)

2 Many thanks to Richard Hayden and Anton Stefanek for their expertise with pgf and pgfplots and their help with this presentation. They also did a substantial portion of the research!