Scalable Performance Analysis of Massively Parallel Stochastic Systems

Richard A. Hayden
rh@doc.ic.ac.uk

Department of Computing, Imperial College London, UK

October 9, 2012
Abstract client/server model

State space grows exponentially as component types are added

Explicit-state analysis techniques do not scale

Simulation is also very costly for large populations
Abstract client/server model

Population CTMC (aggregated) 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
- Simulation is also very costly for large populations
Abstract client/server model

\[ \text{Population CTMC (aggregated) state space:} \]
\[ (N_C(t), N_C^w(t), N_C^p(t), N_S(t), N_S^p(t), N_S^f(t)) \in \mathbb{Z}^6_+ \]

State space grows exponentially as component types are added

Explicit-state analysis techniques do not scale

Simulation is also very costly for large populations
Abstract client/server model

Population CTMC (aggregated) state space:

\[(N_C(t), N_C_w(t), N_C_p(t), N_S(t), N_S_p(t), N_S_f(t)) \in \mathbb{Z}_{+}^6\]
Abstract client/server model

Population CTMC (aggregated) 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
- Simulation is also very costly for large populations
Abstract client/server model

Population CTMC (aggregated) 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
- Simulation is also very costly for large populations
Abstract client/server model

\[
\begin{align*}
\text{Client} & \overset{\text{def}}{=} (\text{req}, \text{rq}).\text{Client}_{\text{wait}} \\
\text{Client}_{\text{wait}} & \overset{\text{def}}{=} (\text{res}, \text{rs}).\text{Client}_{\text{proc}} \\
\text{Client}_{\text{proc}} & \overset{\text{def}}{=} (\text{proc}, \text{rp}).\text{Client}
\end{align*}
\]
Abstract client/server model

Client $\triangleq (\text{req}, r_{rq}) \cdot \text{Client}_{\text{wait}}$

Client$_{\text{wait}} \triangleq (\text{res}, r_{rs}) \cdot \text{Client}_{\text{proc}}$

Client$_{\text{proc}} \triangleq (\text{proc}, r_{p}) \cdot \text{Client}$

Server $\triangleq (\text{req}, r_{rq}) \cdot \text{Server}_{\text{proc}}$

Server$_{\text{proc}} \triangleq (\text{res}, r_{rs}) \cdot \text{Server}$

Server$_{\text{fail}} \triangleq (\text{reset}, r_{rst}) \cdot \text{Server}$

Population CTMC (aggregated) state space:

$(N_C(t), N_C^w(t), N_C^p(t), N_S(t), N_S^p(t), N_S^f(t)) \in \mathbb{Z}^6$
Abstract client/server model

\[
\begin{align*}
\text{Client} & \overset{\text{def}}{=} (\text{req}, \text{r}_{\text{rq}}).\text{Client}_{\text{wait}} \\
\text{Client}_{\text{wait}} & \overset{\text{def}}{=} (\text{res}, \text{r}_{\text{rs}}).\text{Client}_{\text{proc}} \\
\text{Client}_{\text{proc}} & \overset{\text{def}}{=} (\text{proc}, \text{r}_{\text{p}}).\text{Client} \\
\text{Server} & \overset{\text{def}}{=} (\text{req}, \text{r}_{\text{rq}}).\text{Server}_{\text{proc}} \\
& \quad + (\text{fail}, \text{r}_{\text{f}}).\text{Server}_{\text{fail}} \\
\text{Server}_{\text{proc}} & \overset{\text{def}}{=} (\text{res}, \text{r}_{\text{rs}}).\text{Server} \\
\text{Server}_{\text{fail}} & \overset{\text{def}}{=} (\text{reset}, \text{r}_{\text{rst}}).\text{Server} \\
\text{Client}[N_C] & \not\equiv (\text{req}, \text{res}) \not\equiv \text{Server}[N_S]
\end{align*}
\]
Markovian dynamics depend on the aggregate rate of \( \text{req} / \text{res} \)-synchronisations, for example, for \( \text{req} \):

\[
r_{rq \text{min}}(N_C(t), N_S(t))
\]

- e.g. PEPA (‘bounded capacity’), Petri nets, queueing nets

\[
(r_{rq}/N_S)N_C(t)N_S(t)
\]

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

Markovian dynamics depend on the aggregate rate of req/res-synchronisations, for example, for req:

\[ r_{rq} \min(N_C(t), N_S(t)) \]

e.g. PEPA (‘bounded capacity’), Petri nets, queueing nets

\[ (r_{rq}/N_S) N_C(t) N_S(t) \]

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

Population CTMC (aggregated) 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 (aggregated) 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 (aggregated) 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
Population CTMC (aggregated) 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
- Simulation is also very costly for large populations
Mean-field/fluid analysis

- Alleviates the state-space explosion problem for discrete-state Markov models of computer and communication systems
- Derives tractable systems of differential equations approximating statistics of number of components in each local state, e.g.:
Mean-field/fluid analysis

- Alleviates the state-space explosion problem for discrete-state Markov models of computer and communication systems
- Derives tractable systems of differential equations approximating statistics of number of components in each local state, e.g.:
  - **Fluid analysis** of process algebra models[1–5]


Mean-field/fluid analysis

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

- Derives tractable systems of differential equations approximating statistics of number of components in each local state, e.g.:
  
  - **Fluid analysis** of process algebra models
  
  - **Mean-field** analysis of systems of interacting objects[6,7]


Mean-field/fluid analysis

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

- Derives tractable systems of differential equations approximating statistics of number of components in each local state, e.g.:
  - Fluid analysis of process algebra models
  - Mean-field analysis of systems of interacting objects

- These approaches are closely related to and inspired by classical heavy-traffic analysis of queues\[^8\] and analysis of large-scale models in chemistry and biology\[^9\]

---


Mean-field/fluid analysis

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

- Derives tractable systems of differential equations approximating statistics of number of components in each local state, e.g.:
  - Fluid analysis of process algebra models
  - Mean-field analysis of systems of interacting objects

- These approaches are closely related to and inspired by classical heavy-traffic analysis of queues and analysis of large-scale models in chemistry and biology

- Key point: size of approximating system is independent of population size $\Rightarrow$ scalability
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

Passage times
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

Passage times

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

Energy consumption

Both SLAs

SLA H

SLA L

<table>
<thead>
<tr>
<th>Time, t</th>
<th>Faster</th>
<th>Slower</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rewards

<table>
<thead>
<tr>
<th>Time, t</th>
<th>Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Both SLAs

SLA H

SLA L

<table>
<thead>
<tr>
<th>Time, t</th>
<th>Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

- Passage times
- Rewards
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

Passage times

Rewards

![Graph showing Energy Consumption over Time](image)
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

- Passage times
- Rewards

Optimal SLA satisfaction

Optimal SLA satisfaction

Energy Consumption

- Faster
- Slower

Energy consumption

SLA H

SLA L

Both SLAs

N slow

N fast
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

Passage times

Rewards

Optimal SLA satisfaction
Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

- Moment ODEs
- Passage times
- Rewards

Optimal SLA satisfaction

Energy consumption

SLA H
SLA L
Both SLAs

N_{slow}
N_{fast}

0 10 20 30 0 4 8 12
Time, t

0
10
20
50
20
44
60
15
20
Both SLAs
N_{slow} N_{fast}
Energy consumption

SLA H
SLA L

0 10 20 50 20
44
60
15
20
Both SLAs
N_{slow} N_{fast}
Energy consumption

SLA H
SLA L

0 10 20 30 0 4 8 12
Time, t

0
10
20
50
20
44
60
15
20
Both SLAs
N_{slow} N_{fast}
Energy consumption

SLA H
SLA L

Overview

Massively-parallel Markov models

- Grouped PEPA (simple extension of PEPA)
- Population CTMCs
- Stochastic Petri nets, queueing networks

Grouped PEPA analyser (GPA)

Moment ODEs

Passage times

Rewards

Optimal SLA satisfaction

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

Faster
Slower

Time, \( t \)

Probability

Energy Consumption

SLA H
SLA L

Both SLAs

Energy consumption

Both SLAs

Energy consumption

Both SLAs

Rewards

Optimal SLA satisfaction

Faster
Slower

Time, \( t \)
Client/server model fluid limit

\[ \text{Server}_{\text{fail}} \quad \text{fail} @ r_f \]

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

\[ \text{Client}_{\text{wait}} \quad \text{req} \quad \text{res} \]

\[ \text{Client}_{\text{proc}} \quad \text{proc} @ r_p \]

\[ \text{req} \min(N_C(t), N_S(t)) \]

\[ \text{res} \min(N_{Cw}(t), N_{Sp}(t)) \]

\[ N_C \]

\[ N_S \]
Decouple the component count states (limiting independence assumption)
Decouple the component count states (limiting independence assumption)

Construct differential equations by balancing the aggregate rates of Markovian transitions
Client/server model fluid limit

- Decouple the component count states (limiting independence assumption)
- Construct differential equations by balancing the aggregate rates of Markovian transitions
- Deterministic approximations:
  \[(v_C(t), v_{C_w}(t), v_{C_p}(t), v_S(t), v_{S_p}(t), v_{S_f}(t)) \in \mathbb{R}_+^6\]
Client/server model fluid limit

\[
\frac{dv_C(t)}{dt} = \]

\[
\frac{dv_S(t)}{dt} = \]

\[
\text{proc} @ r_p
\]

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

\[
\text{Server}_{\text{proc}} \rightarrow \text{Server} \quad \text{fail} @ r_f
\]

\[
\text{Server} \quad \text{req} \rightarrow \text{Server}_{\text{proc}} \quad \text{res}
\]

\[
\text{Client}_{\text{proc}} \quad \text{res} \rightarrow \text{Client}_{\text{wait}} \quad \text{req}
\]

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

\[
r_s \min(N_{Cw}(t), N_{Sp}(t))
\]

\[
NC
\]

\[
NS
\]
Client/server model fluid limit

\[
\frac{dv_C(t)}{dt} = r_p v_{C_p}(t) - \left( r_q \min(N_C(t), N_S(t)) \right) \quad \text{aggregate rate} \quad \text{Client}_{proc} \rightarrow \text{Client} \\
\frac{dv_S(t)}{dt} = \left( r_q \min(N_C(t), N_S(t)) \right) \quad \text{aggregate rate} \quad \text{Client} \rightarrow \text{Client}_{wait}
\]
Client/server model fluid limit

\[
\frac{dv_C(t)}{dt} = r_p v_{C_p}(t) - r_q \min(v_C(t), v_S(t))
\]

where

- **\(dv_C(t)\)**: aggregate rate from \(Client_{proc} \to Client\)
- **\(r_p v_{C_p}(t)\)**: aggregate rate from \(Client_{proc} \to Client\)
- **\(r_q \min(v_C(t), v_S(t))\)**: aggregate rate from \(Client \to Client_{wait}\)

\[
\frac{dv_S(t)}{dt} = r_{rst} v_{S_f}(t) + r_s \min(v_{C_w}(t), v_{S_p}(t)) - (r_q \min(v_C(t), v_S(t)) + r_f v_S(t))
\]

where

- **\(dv_S(t)\)**: aggregate rate from \(Server \to Server_{proc}, Server \to Server_{fail}\)
- **\(r_{rst} v_{S_f}(t)\)**: aggregate rate from \(Server_{fail} \to Server\)
- **\(r_s \min(v_{C_w}(t), v_{S_p}(t))\)**: aggregate rate from \(Server_{proc} \to Server\)
- **\(r_q \min(v_C(t), v_S(t)) + r_f v_S(t)\)**: aggregate rate from \(Server \to Server_{proc}, Server \to Server_{fail}\)
Client/server model fluid limit

\[ N_C = 10, \quad N_S = 6 \]
Client/server model fluid limit

\[ \text{Server}_{\text{fail}} \]
\[ \text{reset} \circ r_{\text{rst}} \]
\[ \text{fail} \circ r_{f} \]
\[ r_{q} \min(N_{C}(t), N_{S}(t)) \]
\[ r_{s} \min(N_{C_{w}}(t), N_{S_{p}}(t)) \]

\[ N_{C} = 50, N_{S} = 30 \]
Client/server model fluid limit

\[ \text{Server} \left\{ \begin{align*}
\text{proc} & \quad @ \quad r_p \\
& \quad \text{req} \\
\text{Server}_{\text{proc}} & \quad \text{res} \\
\text{Server}_{\text{fail}} & \quad \text{fail} \quad @ \quad r_f \\
\text{Client}_{\text{proc}} & \quad \text{res} \\
\text{Client}_{\text{wait}} & \quad \text{req} \\
\text{Client} & \quad \text{proc} \\
\end{align*} \right. \]

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

\[ r_s \min(N_{C_{\text{w}}}(t), N_{S_{\text{p}}}(t)) \]

\[ N_C = 100, \quad N_S = 60 \]

\[ \text{Time, } t \]

\[ \text{Rescaled component count} \]

\[ 0, 1, 2, 3, 4, 5, 6, 7 \]

\[ 0, 0.2, 0.4, 0.6, 0.8 \]
Client/server model fluid limit

\[ N_C = 200, N_S = 120 \]
Client/server model fluid limit

\[ N_C = 500, \; N_S = 300 \]
Client/server model fluid limit

\[ \text{Client} \]

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

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

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

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

\[ \begin{align*}
    \text{req} & @ r_p \\
    \text{res} & @ \text{req} \\
    \text{req} & @ \text{res} \\
    \text{res} & @ \text{req} \\
    \text{reset} & @ \text{fail} \\
    \text{fail} & @ \text{reset} \\
\end{align*} \]

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

\[ r_s \min(N_{Cw}(t), N_{Sp}(t)) \]

\[ N_C = 1000, \ N_S = 600 \]

\[ \begin{align*}
    N_C & = 1000, \ N_S & = 600 \\
\end{align*} \]
Client/server model fluid limit

\[ \text{Client} \xrightarrow{\text{req}} \text{Client}_{\text{wait}} \xrightarrow{\text{res}} \text{Client}_{\text{proc}} \]

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

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

\[ \text{proc} @ r_p \]

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

\[ \text{reset} @ r_{\text{rst}} \]

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

\[ r_s \min(N_{C_w}(t), N_{S_p}(t)) \]

\[ N_C = 5000, N_S = 3000 \]
Client/server model fluid limit

$$\text{Server}_{\text{fail}} \xrightarrow{\text{fail} \@ r_f} \text{Server}_{\text{proc}}$$

$$\text{reset} \@ r_{\text{rst}}$$

$$\text{proc} @ r_p$$

$$\text{Client}_{\text{wait}} \xrightarrow{\text{req}} \text{Client}_{\text{proc}}$$

$$\text{Client}_{\text{proc}} \xrightarrow{\text{res}} \text{Client}_{\text{wait}}$$

$$\text{Server}_{\text{proc}} \xrightarrow{\text{req}} \text{Server}$$

$$\text{Server} \xrightarrow{\text{res}} \text{Server}_{\text{proc}}$$

$$r_q \min(N_C(t), N_S(t))$$

$$r_s \min(N_{Cw}(t), N_{Sp}(t))$$

$$N_C = 10000, N_S = 6000$$

![Graph](image-url)
Client/server model first-moment approximation

\[ \text{proc} \@ r_p \]

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

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

\[ r_{\text{rq}} \min(N_C(t), N_S(t)) \]

\[ r_{\text{rs}} \min(N_{C_w}(t), N_{S_p}(t)) \]

\[ N_C \]

\[ N_S \]
Client/server model first-moment approximation

Alternatively, can derive the same system of ODEs as approximations to the mean component counts:

\[
(E[N_C(t)], E[N_{Cw}(t)], E[N_{Cp}(t)], E[N_S(t)], E[N_{Sp}(t)], E[N_{Sf}(t)]) \in \mathbb{R}_+^6
\]
Alternatively, can derive the same system of ODEs as approximations to the mean component counts:

\[
\begin{align*}
\mathbb{E}[N_C(t)], \mathbb{E}[N_{Cw}(t)], \mathbb{E}[N_{Cp}(t)], \mathbb{E}[N_S(t)], \mathbb{E}[N_{Sp}(t)], \mathbb{E}[N_{Sf}(t)] \in \mathbb{R}^6
\end{align*}
\]

Convergence as population size increases is much faster
Client/server model first-moment approximation

By the Markov (memoryless) property, we have:

$$\mathbb{P}\{N_C(t + \delta t) - N_C(t) = 1 \mid N(t)\} = r_p N_{Cp}(t) \delta t + o(\delta t)$$
Client/server model first-moment approximation

By the Markov (memoryless) property, we have:

\[
P\{N_C(t + \delta t) - N_C(t) = 1 \mid N(t)\} = r_p N_{Cp}(t) \delta t + o(\delta t)
\]

\[
P\{N_C(t + \delta t) - N_C(t) = -1 \mid N(t)\} = r_q \min(N_C(t), N_S(t)) \delta t + o(\delta t)
\]
By the Markov (memoryless) property, we have:

\[
\begin{align*}
\mathbb{P}\{N_C(t + \delta t) - N_C(t) = 1 \mid N(t)\} &= r_p N_{Cp}(t) \delta t + o(\delta t) \\
\mathbb{P}\{N_C(t + \delta t) - N_C(t) = -1 \mid N(t)\} &= r_q \min(N_C(t), N_S(t)) \delta t + o(\delta t)
\end{align*}
\]

Therefore:

\[
\frac{\mathbb{E}[N_C(t + \delta t) - N_C(t)]}{\delta t} = r_p \mathbb{E}[N_{Cp}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))] + \frac{o(\delta t)}{\delta t}
\]
By the Markov (memoryless) property, we have:

\[ \mathbb{P}\{N_C(t + \delta t) - N_C(t) = 1\mid N(t)\} = r_p N_{C_p}(t) \delta t + o(\delta t) \]
\[ \mathbb{P}\{N_C(t + \delta t) - N_C(t) = -1\mid N(t)\} = r_q \min(N_C(t), N_S(t)) \delta t + o(\delta t) \]

Therefore:

\[ \frac{\mathbb{E}[N_C(t + \delta t) - N_C(t)]}{\delta t} = r_p \mathbb{E}[N_{C_p}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))] + \frac{o(\delta t)}{\delta t} \]

And we recover the differential equation:

\[ \frac{d\mathbb{E}[N_C(t)]}{dt} = r_p \mathbb{E}[N_{C_p}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))] \]
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_C(t)]}{dt} = r_p \mathbb{E}[N_{C_p}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))]\]
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_C(t)]}{dt} = r_p \mathbb{E}[N_{Cp}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))]
\]
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_C(t)]}{dt} = r_p \mathbb{E}[N_{Cp}(t)] - r_q \mathbb{E}[\min(N_C(t), N_S(t))] 
\approx r_p \mathbb{E}[N_{Cp}(t)] - r_q \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)])
\]
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_C(t)]}{dt} = r_p \mathbb{E}[N_{C_p}(t)] - r_q \mathbb{E}\left[\min(N_C(t), N_S(t))\right] \\
\approx r_p \mathbb{E}[N_{C_p}(t)] - r_q \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) \\
\frac{dv_C(t)}{dt} = r_p v_{C_p}(t) - r_q \min(v_C(t), v_S(t)) \\
\]  

\((v(t) \approx \mathbb{E}[N(t)])\)
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_S(t)]}{dt} = r_{rst}\mathbb{E}[N_{Sf}(t)] + r_{rs}\mathbb{E}[\min(N_{Cw}(t), N_{Sp}(t))] \\
- (r_{rq}\mathbb{E}[\min(N_C(t), N_S(t))] + r_{f}\mathbb{E}[N_S(t)])
\]
Client/server model first-moment approximation

\[ \frac{d\mathbb{E}[N_S(t)]}{dt} = r_{rst}\mathbb{E}[N_{S_f}(t)] + r_s\mathbb{E}[\min(N_{C_w}(t), N_{S_p}(t))] \\
- (r_{rq}\mathbb{E}[\min(N_C(t), N_S(t))] + r_f\mathbb{E}[N_S(t)]) \]
Client/server model first-moment approximation

\[
\frac{d\mathbb{E}[N_S(t)]}{dt} = r_{st}\mathbb{E}[N_{S_f}(t)] + r_{s}\mathbb{E}[\min(N_{C_w}(t), N_{S_p}(t))] \\
- (r_{rq}\mathbb{E}[\min(N_C(t), N_S(t))] + r_{f}\mathbb{E}[N_S(t)]) \\
\approx r_{st}\mathbb{E}[N_{S_f}(t)] + r_{s}\min(\mathbb{E}[N_{C_w}(t)], \mathbb{E}[N_{S_p}(t)]) \\
- (r_{rq}\min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) + r_{f}\mathbb{E}[N_S(t)])
\]
Client/server model first-moment approximation

\[ \frac{d\mathbb{E}[N_S(t)]}{dt} = r_{rst}\mathbb{E}[N_{S_f}(t)] + r_s\mathbb{E}[\min(N_{C_w}(t), N_{S_p}(t))] \\
- (r_{rq}\mathbb{E}[\min(N_C(t), N_S(t))] + r_f\mathbb{E}[N_S(t)]) \\
\approx r_{rst}\mathbb{E}[N_{S_f}(t)] + r_s \min(\mathbb{E}[N_{C_w}(t)], \mathbb{E}[N_{S_p}(t)]) \\
- (r_{rq} \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) + r_f\mathbb{E}[N_S(t)]) \]

\[ \frac{dv_S(t)}{dt} = r_{rst}v_{S_f}(t) + r_s \min(v_{C_w}(t), v_{S_p}(t)) \\
- (r_{rq} \min(v_C(t), v_S(t)) + r_f v_S(t)) \quad (v(t) \approx \mathbb{E}[N(t)]) \]
Client/server model first-moment approximation

\[ \text{Client} \xrightarrow{\text{req}} \text{Client}_{\text{wait}} \xrightarrow{\text{res}} \text{Client}_{\text{proc}} \]

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

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

\[ \text{proc} \xrightarrow{r_p} \]

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

\[ r_s \min(N_{Cw}(t), N_{Sp}(t)) \]

\[ N_C = 10, N_S = 6 \]
Client/server model first-moment approximation

\[ \text{proc} @ r_p \]
\[ \text{Client} \]
\[ \text{Client}_{\text{wait}} \]
\[ \text{Client}_{\text{proc}} \]
\[ r_q \min(N_C(t), N_S(t)) \]
\[ r_s \min(N_{Cw}(t), N_{Sp}(t)) \]
\[ N_C = 20, N_S = 12 \]
Client/server model first-moment approximation

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

\[ r_{rs} \min(N_{Cw}(t), N_{Sp}(t)) \]

\[ N_C = 50, N_S = 30 \]
Client/server model first-moment approximation

\[ Server_{\text{fail}} \]
\[ reset @ r_{\text{rst}} \]
\[ fail @ r_{f} \]

\[ Server_{\text{proc}} \]
\[ req \]
\[ res \]

\[ Client_{\text{proc}} \]
\[ req \]
\[ res \]

\[ Client_{\text{wait}} \]
\[ proc @ r_{p} \]

\[ N_{C} \]
\[ N_{S} \]

\[ r_{q} \min(N_{C}(t), N_{S}(t)) \]
\[ r_{s} \min(N_{C_{w}}(t), N_{S_{p}}(t)) \]

\[ N_{C} = 100, N_{S} = 60 \]
Client/server model first-moment approximation

\[
\begin{align*}
N_C &= 500, \quad N_S = 300
\end{align*}
\]
Client/server model second-moment approximation

\[
\begin{align*}
\text{Client} & \quad \text{Server}_\text{fail} \\
\text{Client}_{\text{wait}} & \quad \text{Server} \\
\text{Client}_{\text{proc}} & \quad \text{Server}_{\text{proc}}
\end{align*}
\]
Client/server model second-moment approximation

We can take this further and extend the system of ODEs to also approximate higher-order moments, e.g.:

$$\mathbb{E}[N_C^2(t)] \quad \text{Var}[N_S^2(t)] \quad \mathbb{E}[N_{Cw}^3(t)] \quad \text{Skew}[N_{Sp}(t)]$$
We can take this further and extend the system of ODEs to also approximate higher-order moments, e.g.:

\[
\mathbb{E}[N_{C}^2(t)] \quad \text{Var}[N_{S}^2(t)] \quad \mathbb{E}[N_{Cw}^3(t)] \quad \text{Skew}[N_{Sp}(t)]
\]

- Gives us more detailed information about the distribution of the component counts
Again by the Markov (memoryless) property, we have:

\[
\mathbb{P}\{N_C^2(t + \delta t) - N_C^2(t) = 2N_C(t) + 1 \mid N(t)\} = r_p N_{Cp}(t)\delta t + o(\delta t)
\]
Again by the Markov (memoryless) property, we have:

\[
P\{N_C^2(t + \delta t) - N_C^2(t) = 2N_C(t) + 1 \mid N(t)\} = r_p N_{Cp}(t)\delta t + o(\delta t)
\]

\[
P\{N_C^2(t + \delta t) - N_C^2(t) = -2N_C(t) + 1 \mid N(t)\} = r_q \min(N_C(t), N_S(t))\delta t + o(\delta t)
\]
Again by the Markov (memoryless) property, we have:

\[
P\{N_C^2(t + \delta t) - N_C^2(t) = 2N_C(t) + 1 \mid N(t)\} = r_p N_{Cp}(t) \delta t + o(\delta t)
\]

\[
P\{N_C^2(t + \delta t) - N_C^2(t) = -2N_C(t) + 1 \mid N(t)\} = r_q \min(N_C(t), N_S(t)) \delta t + o(\delta t)
\]

\[
\frac{\mathbb{E}[N_C^2(t + \delta t) - N_C^2(t)]}{\delta t} = r_p \mathbb{E}[(2N_C(t) + 1)N_{Cp}(t)]
\]

\[
- r_q \mathbb{E}[(2N_C(t) - 1) \min(N_C(t), N_S(t))] + \frac{o(\delta t)}{\delta t}
\]
Again by the Markov (memoryless) property, we have:

\[
\Pr\{N_C^2(t + \delta t) - N_C^2(t) = 2N_C(t) + 1 | N(t)\} = r_p N_{Cp}(t) \delta t + o(\delta t)
\]

\[
\Pr\{N_C^2(t + \delta t) - N_C^2(t) = -2N_C(t) + 1 | N(t)\} = r_q \min(N_C(t), N_S(t)) \delta t + o(\delta t)
\]

\[
\frac{\mathbb{E}[N_C^2(t + \delta t) - N_C^2(t)]}{\delta t} = r_p \mathbb{E}[(2N_C(t) + 1)N_{Cp}(t)]
\]

\[
- r_q \mathbb{E}[(2N_C(t) - 1) \min(N_C(t), N_S(t))] + \frac{o(\delta t)}{\delta t}
\]

\[
\frac{d\mathbb{E}[N_C^2(t)]}{dt} = 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)]
\]

\[
- 2r_q \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + r_q \mathbb{E}[\min(N_C(t), N_S(t))]
\]
Client/server model second-moment approximation

\[
\frac{d\mathbb{E}[N^2_C(t)]}{dt} = 2r_p\mathbb{E}[N_C(t)N_{Cp}(t)] + r_p\mathbb{E}[N_{Cp}(t)] - 2r_q\mathbb{E}[N_C(t)\min(N_C(t), N_S(t))] + r_q\mathbb{E}[\min(N_C(t), N_S(t))]
\]
Client/server model second-moment approximation

\[ \frac{d\mathbb{E}[N_C^2(t)]}{dt} = 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)] \\
- 2r_{rq} \mathbb{E}[N_C(t)\min(N_C(t), N_S(t))] + r_{rq} \mathbb{E}[\min(N_C(t), N_S(t))] \]
Client/server model second-moment approximation

\[
\frac{d\mathbb{E}[N_C^2(t)]}{dt} = 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)] - 2r_{rq} \mathbb{E}[N_C(t) \min(N_C(t), N_S(t))] + r_{rq} \mathbb{E}[\min(N_C(t), N_S(t))] \\
\approx 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)] - 2r_{rq} \min(\mathbb{E}[N_C^2(t)], \mathbb{E}[N_C(t)N_S(t)]) + r_{rq} \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)])
\]
Client/server model second-moment approximation

\[
\frac{d\mathbb{E}[N_C^2(t)]}{dt} = 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)]
- 2r_q \mathbb{E}[N_C(t)\min(N_C(t), N_S(t))] + r_q \mathbb{E}[\min(N_C(t), N_S(t))]
\approx 2r_p \mathbb{E}[N_C(t)N_{Cp}(t)] + r_p \mathbb{E}[N_{Cp}(t)]
- 2r_q \min(\mathbb{E}[N_C^2(t)], \mathbb{E}[N_C(t)N_S(t)]) + r_q \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)])
\]

\[
\frac{dv_{C2}(t)}{dt} = 2r_p v_{CCp}(t) + r_p v_{Cp}(t)
- 2r_q \min(v_{C2}(t), v_{CS}(t)) + r_q \min(v_C(t), v_S(t))
\]
Client/server model second-moment approximation

\[ N_C = 50, \quad N_S = 30 \]
Client/server model second-moment approximation

\[ N_C = 100, \quad N_S = 60 \]
Client/server model second-moment approximation

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

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

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

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

\[ \text{Server} \]

\[ \text{reset} @ r_{\text{rst}} \]

\[ \text{fail} @ r_{f} \]

\[ \text{proc} @ r_{p} \]

\[ r_{q} \min(N_{C}(t), N_{S}(t)) \]

\[ r_{rs} \min(N_{Cw}(t), N_{Sp}(t)) \]

\[ N_{C} = 200, N_{S} = 120 \]
Client/server model second-moment approximation

![Diagram of client/server model](image)

\[
N_C = 500, \quad N_S = 300
\]
Client/server model second-moment approximation

\[ \text{Client} \]

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

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

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

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

\[ r_p \]

\[ r_{\text{req}} \min(N_C(t), N_S(t)) \]

\[ r_{\text{res}} \min(N_{C_w}(t), N_{S_p}(t)) \]

\[ N_C = 1000, \; N_S = 600 \]
Client/server model second-moment approximation

\[ N_C = 10000, \quad N_S = 6000 \]
Client/server model skewness approximation

\[ \text{Server} \leftarrow \text{Client} \]

**Client**

1. **Client**
   - req
   - res
   - \( \text{Client}_{\text{wait}} \)
   - \( \text{Client}_{\text{proc}} \)

**Server**

1. **Server**
   - req
   - res
   - \( \text{Server}_{\text{proc}} \)

2. **Server_{fail}**
   - reset @ \( r_{\text{rst}} \)
   - fail @ \( r_{f} \)

**Equations**

\[ \begin{align*}
\text{req} & \, \min(N_C(t), N_S(t)) \\
\text{res} & \, \min(N_{C_{\text{w}}}(t), N_{S_{\text{p}}}(t))
\end{align*} \]
Client/server model skewness approximation

\[ \text{Server}_{\text{fail}} \rightarrow \text{reset} @ r_{\text{rst}} \quad \text{fail} @ r_{\text{f}} \]

\[ \text{proc} @ r_{\text{p}} \]

\[ \text{Client}_{\text{wait}} \rightarrow \text{req} \quad \text{res} \]

\[ \text{Client}_{\text{proc}} \rightarrow \text{req} \quad \text{res} \]

\[ r_{\text{q}} \min(N_{\text{C}}(t), N_{\text{S}}(t)) \]

\[ r_{\text{rs}} \min(N_{\text{Cw}}(t), N_{\text{Sp}}(t)) \]

\[ N_{\text{C}} = 100, \quad N_{\text{S}} = 60 \]

\[ \begin{align*}
    \text{Client} & \quad \text{Client}_{\text{wait}} \\
    \text{Client}_{\text{proc}} & \quad \text{Server}_{\text{proc}} \\
    \text{Server}_{\text{fail}} & \quad \text{Server}_{\text{fail}}
\end{align*} \]

\[ \begin{align*}
    \text{Std. comp. count skewness} & \quad \text{Std. comp. count skewness}
\end{align*} \]
Switch points

\[
\begin{align*}
N_C &= 100, 
N_S &= 60
\end{align*}
\]
Switch points

$N_C = 100, N_S = 60$
Switch points

\[ N_C = 100, \; N_S = 60 \]
Switch points

\[
\mathbb{E}\left[\min(N_C(t), N_S(t))\right] \approx \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)])
\]
Switch points

\[ \mathbb{E}[\min(N_C(t), N_S(t))] \approx \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) \]

- Most accurate when \( N_C(t) \) and \( N_S(t) \) are unlikely to be close
Switch points

\[ \mathbb{E}[\min(N_C(t), N_S(t))] \approx \min(\mathbb{E}[N_C(t)], \mathbb{E}[N_S(t)]) \]

- Most accurate when \( N_C(t) \) and \( N_S(t) \) are unlikely to be close
- However, often we wish to design a system so that they are likely to be close, e.g. quality and efficiency driven regime for queueing\(^{[10]}\)/resource-constrained models

Improving the approximation

- As populations are scaled up, it is known that the component counts become approximately jointly normal before they become deterministic\[^{[11]}\]

Improving the approximation

- As populations are scaled up, it is known that the component counts become approximately jointly normal before they become deterministic.

- **Idea:** compute approximation to $\mathbb{E}[\min(N_C(t), N_S(t))]$ by assuming that $(N_C(t), N_S(t))$ is jointly normal\(^{[12]}\)

---

Improving the approximation

- As populations are scaled up, it is known that the component counts become approximately jointly normal before they become deterministic.
- **Idea:** compute approximation to $\mathbb{E}[\min(N_C(t), N_S(t))]$ by assuming that $(N_C(t), N_S(t))$ is jointly normal.

$$N_C = 100, \quad N_S = 60$$
Improving the approximation

- As populations are scaled up, it is known that the component counts become **approximately jointly normal** before they become deterministic
- **Idea:** compute approximation to $\mathbb{E}[\min(N_C(t), N_S(t))]$ by assuming that $(N_C(t), N_S(t))$ is jointly normal

$$N_C = 100, N_S = 60$$
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:[13]

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:\[^{13}\]
  - **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:[13]
  - 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:\[13\]
  
  - **Individual passage times**: track the time taken for an individual to complete a task
    - *Scalable 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:[13]

  - **Individual passage times**: track the time taken for an individual to complete a task
    - *Scalable 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 moment-derived bounds on CDF*

---

**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 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 moment-derived bounds on CDF

- Can be specified using the *unified stochastic probe framework*[^14], supported by GPA tool[^15]


Individual passage times

- **Individual passage times**: track the time taken for an individual to complete a task
- *Scalable direct approximation to the entire CDF*
Individual passage times

- **Individual passage times**: track the time taken for an individual to complete a task
- **Scalable direct approximation to the entire CDF**
- Formal convergence proofs for increasing population size\[13\]

Individual passage times

- **Individual passage times**: track the time taken for an individual to complete a task
- *Scalable direct approximation to the entire CDF*
- Formal convergence proofs for increasing population size
- Builds on techniques combining *Little’s Law* with fluid analysis to obtain average steady-state passage times\(^3,16,17\)

---


Individual passage times

- **Individual passage times**: track the time taken for an individual to complete a task
- *Scalable direct approximation to the entire CDF*
- Formal convergence proofs for increasing population size
- Builds on techniques combining *Little’s Law* with fluid analysis to obtain *average steady-state passage times*
- Recently, a similar approach has been applied to performance queries specified using a subset of *Continuous Stochastic Logic (CSL)*[^18]

Individual passage time example

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

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 time example

\[ L_{\text{Probe}} \overset{\text{def}}{=} \text{proc} : \text{start}, \text{proc} : \text{stop} \]
Individual passage time example

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

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

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

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

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

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

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

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

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

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

\[
\mathbb{P}\{ T \leq t \} = \mathbb{P}\{ C(t) \in \{ \text{Client}', \text{Client}'_{\text{wait}}, \text{Client}'_{\text{proc}} \} \} \\
= \mathbb{E}[\mathbf{1}\{C(t) = \text{Client}'\}] + \mathbb{E}[\mathbf{1}\{C(t) = \text{Client}'_{\text{wait}}\}] + \mathbb{E}[\mathbf{1}\{C(t) = \text{Client}'_{\text{proc}}\}] \\
= \mathbb{E}[N_{\text{Client}'}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{wait}}}(t)] + \mathbb{E}[N_{\text{Client}'_{\text{proc}}}(t)]
\]

\[
\mathbb{P}\{ T \leq t \} \approx \nu_{\text{Client}'}(t) + \nu_{\text{Client}'_{\text{wait}}}(t) + \nu_{\text{Client}'_{\text{proc}}}(t)
\]
Individual passage time example

![Graph showing the fluid approximation of probability over time. The graph plots probability on the y-axis and time on the x-axis, with a curve indicating an increasing probability as time progresses.]
Individual passage time example

$N_C = 10$, $N_S = 6$
Individual passage time example

\[ N_C = 20, \quad N_S = 12 \]
Individual passage time example

$N_C = 50, N_S = 30$
Individual passage time example

$N_C = 100, N_S = 60$
Individual passage time example

$N_C = 200, N_S = 120$
Global passage time example

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

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 time example

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

\[ LProbe \overset{\text{def}}{=} proc : \text{start}_{\text{local}} \]

\[ \text{GProbe} \overset{\text{def}}{=} \epsilon : \text{start}, \text{start}_{\text{local}}[N_C/2] : \text{stop} \]
Global passage time example

\[ T := \inf\{t \geq 0 : N_{C'}(t) + N_{C'_w}(t) + N_{C'_p}(t) \geq N_C/2\} \]
Global passage time example

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

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 time example
Global passage time example

$N_C = 10, N_S = 6$

- **Point-mass approximation**
- **Actual CDF (simulation)**

![Graph showing the comparison between the point-mass approximation and the actual CDF (simulation) for passage time with $N_C = 10$ and $N_S = 6$. The graph plots probability against time, with a dotted line indicating the point of comparison.]
Global passage time example

\( N_C = 20, N_S = 12 \)
Global passage time example

\[ N_C = 50, N_S = 30 \]
Global passage time example

\[ N_C = 300, N_S = 120 \]
Global passage time example

\[ N_C = 500, N_S = 300 \]
Global passage time example

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 $T$
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[19]

Global passage time bounds — first moments

$N_C = 20, N_S = 12$

**Half of the clients:**

**Three quarters of the clients:**

**All of the clients:**
Global passage time bounds — higher moments

![Graph showing CDFs of different orders]

- **1st order**
- **2nd order**
- **4th order**
- **Actual CDF**
Scalable analysis of accumulated reward measures
Accumulated reward measures

- Cost, energy, heat, …
- Constant rate
Accumulated reward measures

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

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

\[ \text{Total energy}(t) = r_{S} \int_{0}^{t} N_{S}(u) \, du + r_{Sp} \int_{0}^{t} N_{Sp}(u) \, du \]
Accumulated reward measures

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

\[
\text{Server}_\text{proc} \quad \text{Server}_\text{proc} \quad \int_0^t \text{r}_\text{Sp}(u) \, du + \int_0^t \text{r}_\text{Sp}(u) \, du
\]

![Graph showing state at t, Count, and N_s(t)]
Accumulated reward measures

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

\[
\text{total energy}(t) = \int_0^t r_s N_s(u) \, du + \int_0^t r_{sp} N_{sp}(u) \, du
\]
Accumulated reward measures

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

\[
\text{Server}_{\text{proc}} \quad r_{Sp}
\]

\[
\text{Server} \quad r_S
\]

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

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

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

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

**Server**

**Server\_proc**

**Server\_fail**
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

- Explicit state analysis and simulation are also very costly for accumulated rewards
Moment approximations of accumulated rewards

- Explicit state analysis and simulation are also very costly for accumulated rewards

- Can extend the ODE system for component count moments with ODEs for moments of accumulated counts:[20]

---

Moment approximations of accumulated rewards

- Explicit state analysis and simulation are also very costly for accumulated rewards

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

\[
\frac{d}{dt} \mathbb{E} \left[ \int_0^t N_{sp}(u) \, du \right] = \cdots
\]
Moment approximations of accumulated rewards

- Explicit state analysis and simulation are also very costly for accumulated rewards

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

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

- Can be used, for example, to derive completion time distributions
Moment approximations of accumulated rewards

- Explicit state analysis and simulation are also very costly for accumulated rewards

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

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

- Can be used, for example, to derive completion time distributions
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

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

\[
\frac{d}{dt} \mathbb{E} \left[ \left( \int_{0}^{t} N_S(u) \, du \right)^2 \right] =
\]

First-order moments

\[
\frac{d}{dt} \mathbb{E}[N_S(t)] = \cdots
\]
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**

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

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

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

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

\[ \frac{d}{dt} \mathbb{E} \left[ N_S(t) \int_0^t N_S(u) \, du \right] = \mathbb{E} \left[ N_C(t) \int_0^t N_S(u) \, du \right] + \cdots + \mathbb{E}[N_S^2(t)] \]
Trade-off between energy and performance
Trade-off between energy and performance

Client

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

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

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

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

\[ \text{proc} \]

\[ \text{req} \]

\[ \text{res} \]

\[ \text{reset} \]

\[ \text{fail} \]

\[ N_C \]

\[ N_S \]

\begin{figure}
\centering
\includegraphics[width=\textwidth]{trade-off}
\caption{Trade-off between energy and performance}
\end{figure}

\begin{figure}[h]
\centering
\begin{subfigure}{0.49\textwidth}
\centering
\begin{tikzpicture}[node distance=2cm]
\node (client) {Client\_wait};
\node (clientproc) [below of=client] {Client\_proc};
\node (server) [right of=clientproc] {Server\_proc};
\node (serverfail) [above of=server] {Server\_fail};

\draw [->] (client) -- (clientproc) node [midway, left] {res};
\draw [->] (client) -- (server) node [midway, right] {req};
\draw [->] (clientproc) -- (server) node [midway, above] {res};
\draw [->] (server) -- (serverfail) node [midway, right] {fail};
\draw [->] (clientproc) -- (serverfail) node [midway, left] {reset};

\node at (current bounding box.north) {\( N_C \)};
\end{tikzpicture}
\end{subfigure}
\begin{subfigure}{0.49\textwidth}
\centering
\begin{tikzpicture}[node distance=2cm]
\node (client) {Client\_wait};
\node (clientproc) [below of=client] {Client\_proc};
\node (server) [right of=clientproc] {Server\_proc};
\node (serverfail) [above of=server] {Server\_fail};

\draw [->] (client) -- (clientproc) node [midway, left] {res};
\draw [->] (client) -- (server) node [midway, right] {req};
\draw [->] (clientproc) -- (server) node [midway, above] {res};
\draw [->] (server) -- (serverfail) node [midway, right] {fail};
\draw [->] (clientproc) -- (serverfail) node [midway, left] {reset};

\node at (current bounding box.north) {\( N_S \)};
\end{tikzpicture}
\end{subfigure}
\end{figure}

\begin{figure}[h]
\centering
\begin{subfigure}{0.49\textwidth}
\centering
\begin{axis}[title={Client serviced before \( t \)},xlabel={Time, \( t \)},ylabel={Probability},legend pos=north west]
\addplot[blue] table [x expr=
\thisrow{Time}_t, y=\thisrow{Probability}_t] {data.csv};
\end{axis}
\end{subfigure}
\begin{subfigure}{0.49\textwidth}
\centering
\begin{axis}[title=\( \mathbb{E}[\text{total energy}(t)] \),xlabel={Time, \( t \)},ylabel={Energy},legend pos=north west]
\addplot[blue] table [x expr=
\thisrow{Time}_t, y=\thisrow{Energy}_t] {data.csv};
\end{axis}
\end{subfigure}
\end{figure}
Trade-off between energy and performance

![Diagram of client-server interaction with energy and probability graphs]

- Client
  - req
  - Client\text{\_wait}
  - res
  - Client\text{\_proc}
  - \text{\textbf{N}}_C

- Server
  - req
  - Server\text{\_proc}
  - \text{\textbf{N}}_S

- Server\text{\_fail}
  - reset
  - fail

- Energy: \( E_{\text{total energy}(t)} \)
  - \text{\textbf{E}[total energy(t)]}

- Probability: Client serviced before \( t \)
  - SLA
  - 7s
  - \( \geq 0.99 \)

- Time, \( t \):
  - 0, 2, 4, 6, 8, 10
  - 0, 50, 100, 150
Trade-off between energy and performance

- **Client**
  - req
  - res
  - \( \text{Client}_{\text{wait}} \)
  - \( \text{Client}_{\text{proc}} \)

- **Server**
  - req
  - res
  - \( \text{Server}_{\text{proc}} \)
  - \( \text{Server}_{\text{sleep}} \)
  - fail
  - reset

- **Client serviced before**
  - Probability
  - Time, \( t \)
  - SLA \( 7s \) \( \geq 0.99 \)

- **Energy**
  - \( E[\text{total energy}(t)] \)
  - Time, \( t \)

- **Probability**
  - Time, \( t \)
  - \( N_C \)

- **Energy**
  - Time, \( t \)
  - \( N_S \)
Trade-off between energy and performance

- **Client**
  - req
  - Client\(_{\text{wait}}\)
  - res
  - Client\(_{\text{proc}}\)
  - proc

- **Server**
  - req
  - Server\(_{\text{proc}}\)
  - res
  - Server\(_{\text{fail}}\)
  - fail
  - reset
  - sleep/wakeup

- **Ns**
- **NC**

**Graphs:**
- Probability of Client serviced before time, \(t\)
  - SLA: 7s \(\geq 0.99\)
  - Time, \(t\): 0 to 10
- Energy, \(E_{\text{total energy}(t)}\)
  - Expected energy
  - Time, \(t\): 0 to 6
Trade-off between energy and performance

**Scalable analysis** allows exploration of many configurations
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations

- Number of servers, $N_S$
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations

- Number of servers, $N_S$
- Sleep/wakeup rate
Trade-off between energy and performance

Scalable analysis allows exploration of many configurations

- Number of servers, $N_S$
- Sleep/wakeup rate

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

Scalable analysis allows exploration of many configurations

- Number of servers, $N_S$
- Sleep/wakeup rate

Minimise energy consumption while satisfying SLAs

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

**Scalable analysis** allows exploration of many configurations

- Number of servers, $N_S$
- Sleep/wakeup rate

**Minimise** energy consumption while satisfying SLAs

![Graph showing trade-off between energy consumption and sleep rate for different numbers of servers $N_S$. The graph indicates that lower sleep rates correspond to higher energy consumption, and the SLA met condition is satisfied at specific points on this graph.](image)

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

Scalable analysis allows exploration of many configurations

- Number of servers, $N_S$
- Sleep/wakeup 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

- Number of servers, $N_S$
- Sleep/wakeup 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

- Number of servers, $N_S$
- Sleep/wakeup 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

- Number of servers, \( N_S \)
- Sleep/wakeup rate

Minimise energy consumption while satisfying SLAs

Individual passage-time SLA: clients must finish in at most 7s \( \geq 99.5\% \) of the time
Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool\cite{15,21}: http://code.google.com/p/gpanalyser/


Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool: http://code.google.com/p/gpanalyser/

- Supports GPEPA (simple extension of PEPA) or lower level stoichiometric equation population CTMC language
Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool: http://code.google.com/p/gpanalyser/

- Supports GPEPA (simple extension of PEPA) or lower level stoichiometric equation population CTMC language

- Scalable analysis of arbitrary moments of component counts with various moment closure options
Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool: http://code.google.com/p/gpanalyser/

- Supports GPEPA (simple extension of PEPA) or lower level stoichiometric equation population CTMC language

- Scalable analysis of arbitrary moments of component counts with various moment closure options

- Passage-time measures with full regular expression probe syntax
Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool: http://code.google.com/p/gpanalyser/

- Supports GPEPA (simple extension of PEPA) or lower level stoichiometric equation population CTMC language

- Scalable analysis of arbitrary moments of component counts with various moment closure options

- Passage-time measures with full regular expression probe syntax

- Accumulated reward measures
Grouped PEPA Analyser (GPA)

- All of the techniques described here (and more!) implemented in the freely available and open source GPA tool: http://code.google.com/p/gpanalyser/

- Supports GPEPA (simple extension of PEPA) or lower level stoichiometric equation population CTMC language

- Scalable analysis of arbitrary moments of component counts with various moment closure options

- Passage-time measures with full regular expression probe syntax

- Accumulated reward measures

- Solution of global constrained optimisation problems (e.g. minimal energy usage to satisfy SLAs)
Thank you!

... and also to Anton Stefanek and Jeremy Bradley who were involved in this research