Parallel 2012

Parallelität messen und visualisieren

Stefan Ruppert, Manuel Núñez
MyARM GmbH
Altkönigstraße 7
65830 Kriftel
Deutschland
Web: http://www.myarm.com
eMail: info@myarm.com

Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

Inhalt

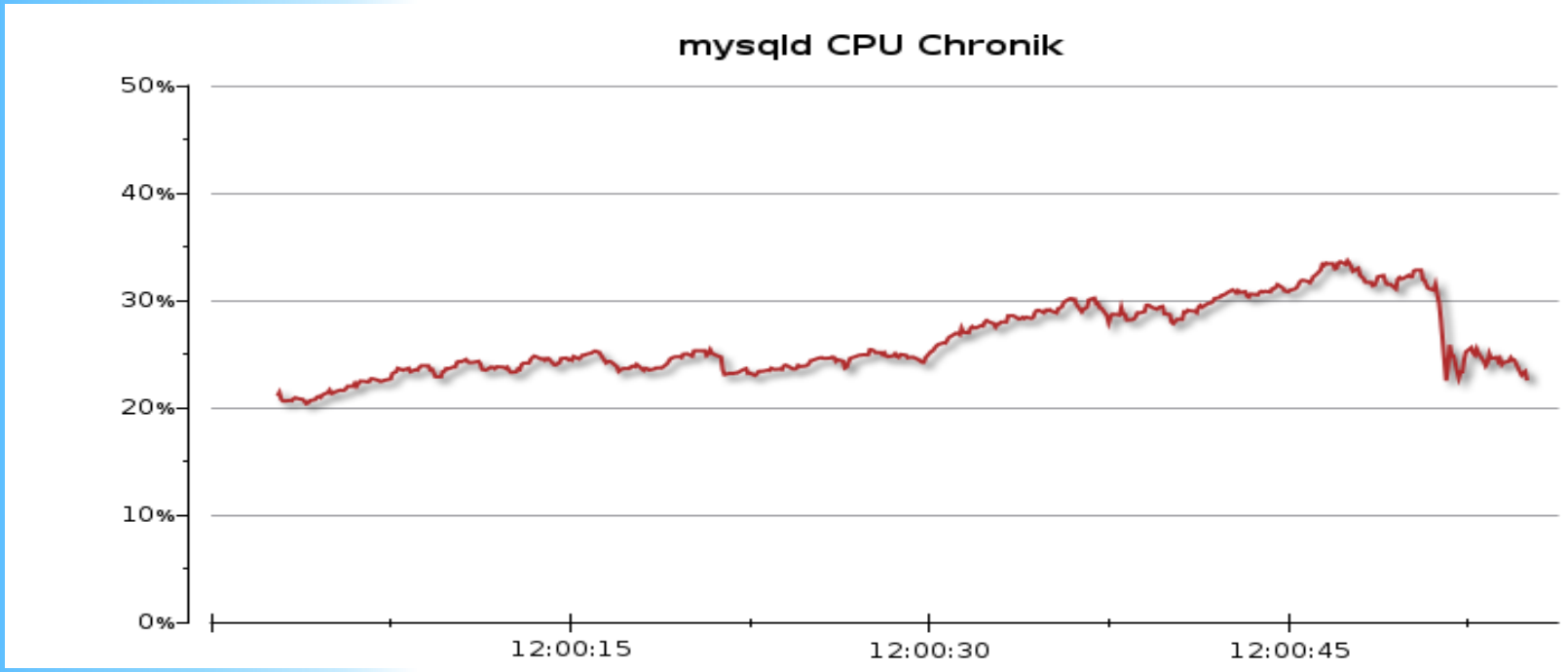
- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

Problemstellung: Messen

- Was soll gemessen werden?
  - Eine Anwendung?

Problemstellung: Messen

- Was soll gemessen werden?
  - Eine Anwendung?

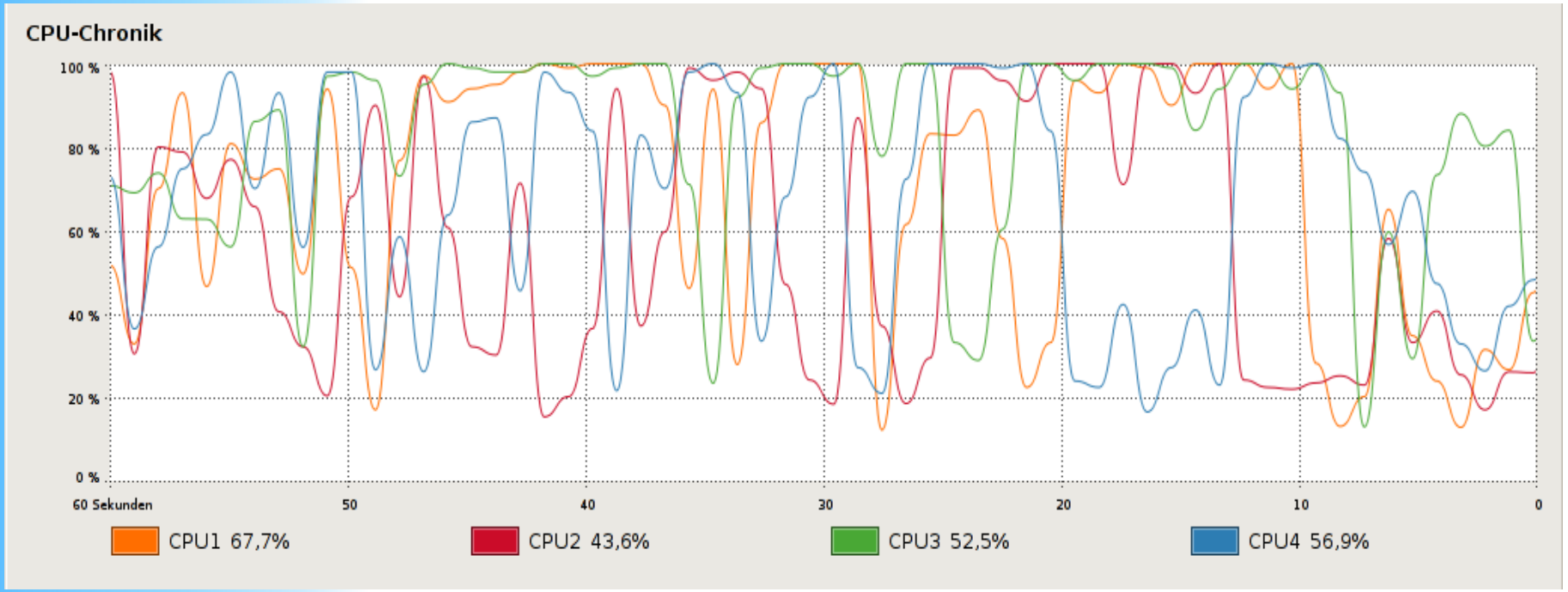


Problemstellung: Messen

- Was soll gemessen werden?
  - Auslastung eines Rechners?

Problemstellung: Messen

- Was soll gemessen werden?
  - Auslastung eines Rechners?

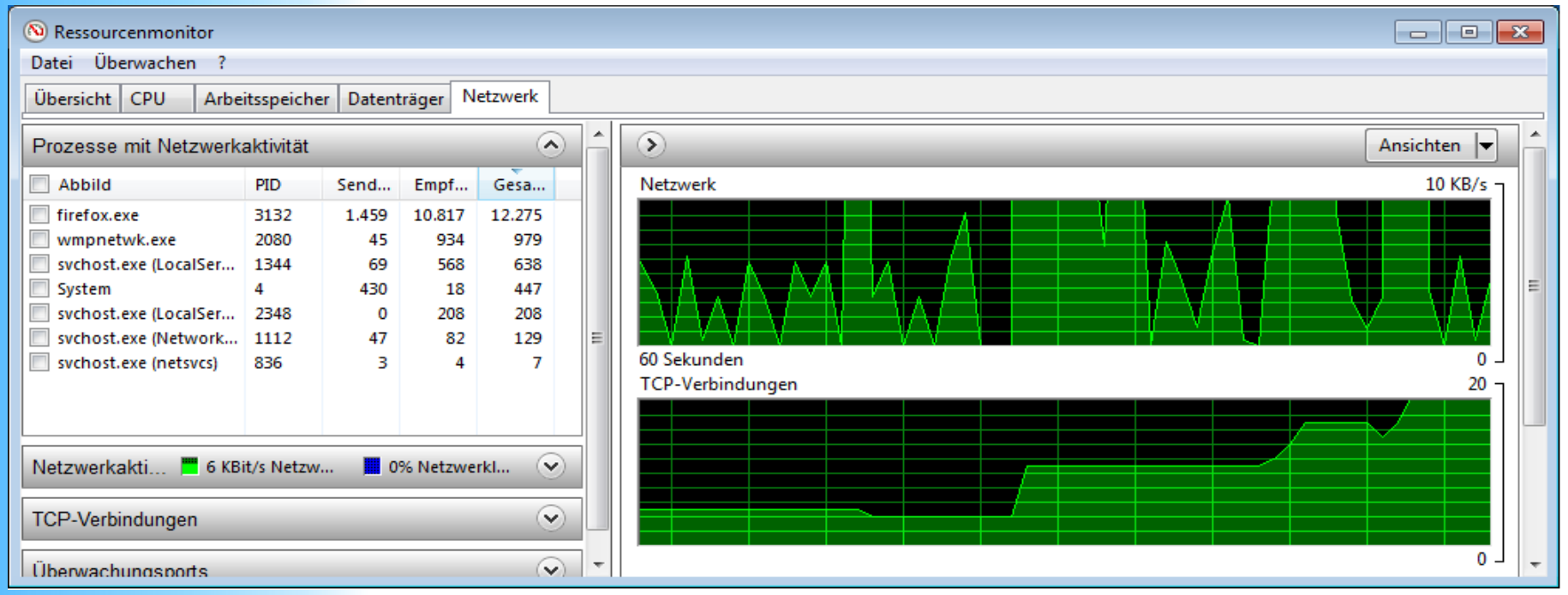


Problemstellung: Messen

- Was soll gemessen werden?
  - Netzwerk-Last bzw. -Latenz?

Problemstellung: Messen

- Was soll gemessen werden?
  - Netzwerk-Last bzw. -Latenz?



Problemstellung: Messen

- Was soll gemessen werden?
  - Auslastung eines Rechners?
  - Eine Anwendung?
  - Netzwerk-Last bzw. -Latenz?
- Am Besten Alles!
  - Aber „Was“ genau?
  - und „Wie“?

Problemstellung: Messen

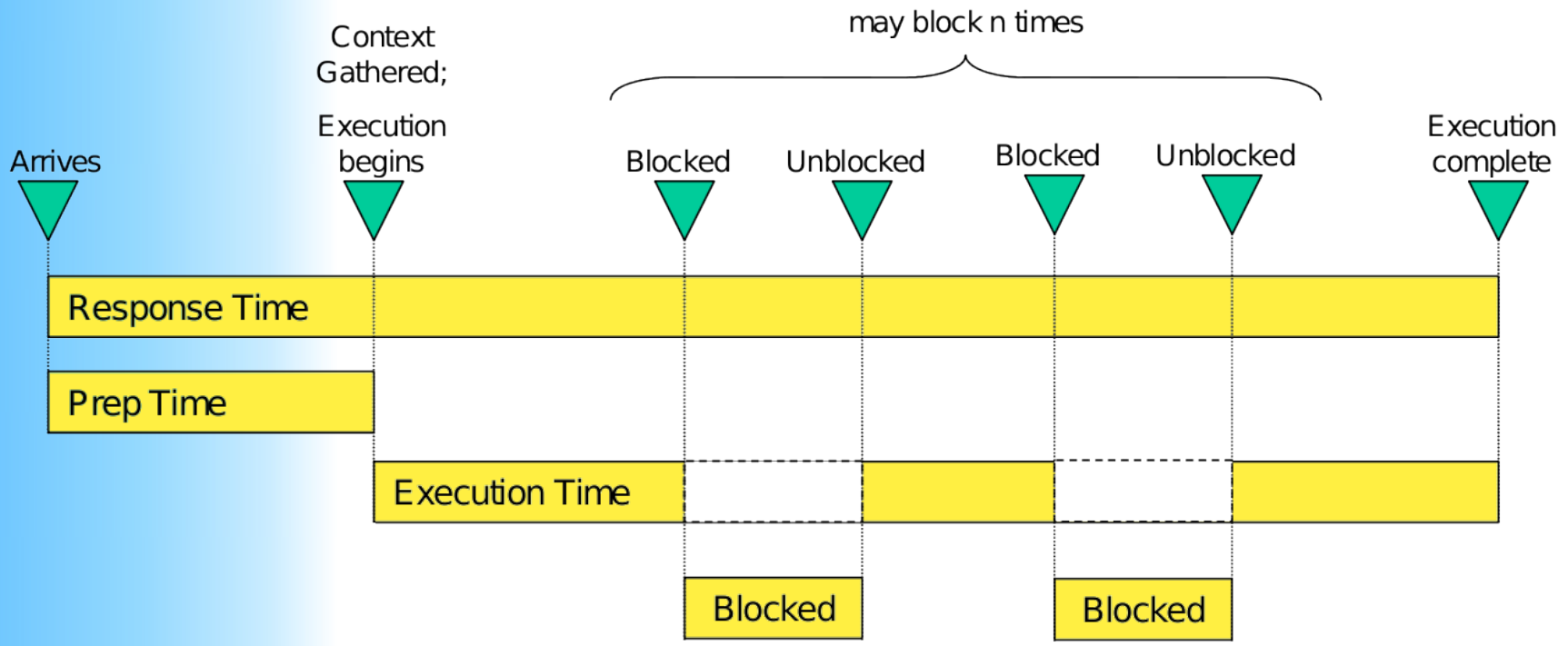
- „Was“ genau soll gemessen werden?
  - Fokusierung auf zentrale Arbeitseinheiten der Anwendung
  - Antwortzeit der Arbeitseinheit enthält alle Einflüsse
    - CPU- und Netzwerk-Last
    - Algorithmik, Komplexität, etc
  - Beispiel: „Laden“ und „Anzeigen“ einer Web-Seite im Web-Browser

Problemstellung: Messen

- Wie?
  - Instrumentierung der Anwendung zur Bestimmung der Antwortzeit
  - Arbeitsheinheit (AE) zum Messen finden und
    - Startzeitpunkt (SZP) der AE festhalten
    - Arbeitseinheit durchführen
    - Endzeitpunkt (EZP) der AE festhalten
    - Antwortzeit= EZP - SZP

Problemstellung: Messen

Antwortzeit



Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

Problemstellung: Visualisieren

- Was soll noch dargestellt werden?
  - Messwerte?
  - Verteilung von Messwerten?
  - Komplexe Messhierarchien?
  - Reduktion auf einfache Kenngrößen?
  - Statistische Kennwerte?

Problemstellung: Visualisieren

- Was soll dargestellt werden?
  - Messwerte?
  - Verteilung von Messwerten?
  - Komplexe Messhierarchien?
  - Reduktion auf einfache Kenngrößen?
  - Statistische Kennwerte?
- Und wie präsentieren?

Problemstellung: Visualisieren

- Textuelle Darstellung?

Identification	AppName	StartTime	Dura...	%	Net ...	%	Status
MandelbrotCalculation	QParallelMandelbrot	12:05:36.401	3040.621	100....	-7075.531	-232.7%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2312.097	76.0%	27.760	0.9%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.414	2275.704	74.8%	2275.704	74.8%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:38.697	8.633	0.3%	8.633	0.3%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2339.393	76.9%	26.094	0.9%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	1095.405	36.0%	1095.405	36.0%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:37.508	1070.427	35.2%	1070.427	35.2%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:38.586	147.467	4.8%	147.467	4.8%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	3040.343	100....	30.684	1.0%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	10.632	0.3%	10.632	0.3%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.435	2999.027	98.6%	2999.027	98.6%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2424.319	79.7%	18.780	0.6%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	192.220	6.3%	192.220	6.3%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.605	2213.319	72.8%	2213.319	72.8%	GOOD

Problemstellung: Visualisieren

- Einfach, aber nicht anschaulich!

Identification	AppName	StartTime	Dura...	%	Net ...	%	Status
MandelbrotCalculation	QParallelMandelbrot	12:05:36.401	3040.621	100...	-7075.531	-232.7%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2312.097	76.0%	27.760	0.9%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.414	2275.704	74.8%	2275.704	74.8%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:38.697	8.633	0.3%	8.633	0.3%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2339.393	76.9%	26.094	0.9%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	1095.405	36.0%	1095.405	36.0%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:37.508	1070.427	35.2%	1070.427	35.2%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:38.586	147.467	4.8%	147.467	4.8%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	3040.343	100....	30.684	1.0%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	10.632	0.3%	10.632	0.3%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.435	2999.027	98.6%	2999.027	98.6%	GOOD
ThreadExecution	QParallelMandelbrot	12:05:36.401	2424.319	79.7%	18.780	0.6%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.405	192.220	6.3%	192.220	6.3%	GOOD
ChunkCalculation	QParallelMandelbrot	12:05:36.605	2213.319	72.8%	2213.319	72.8%	GOOD

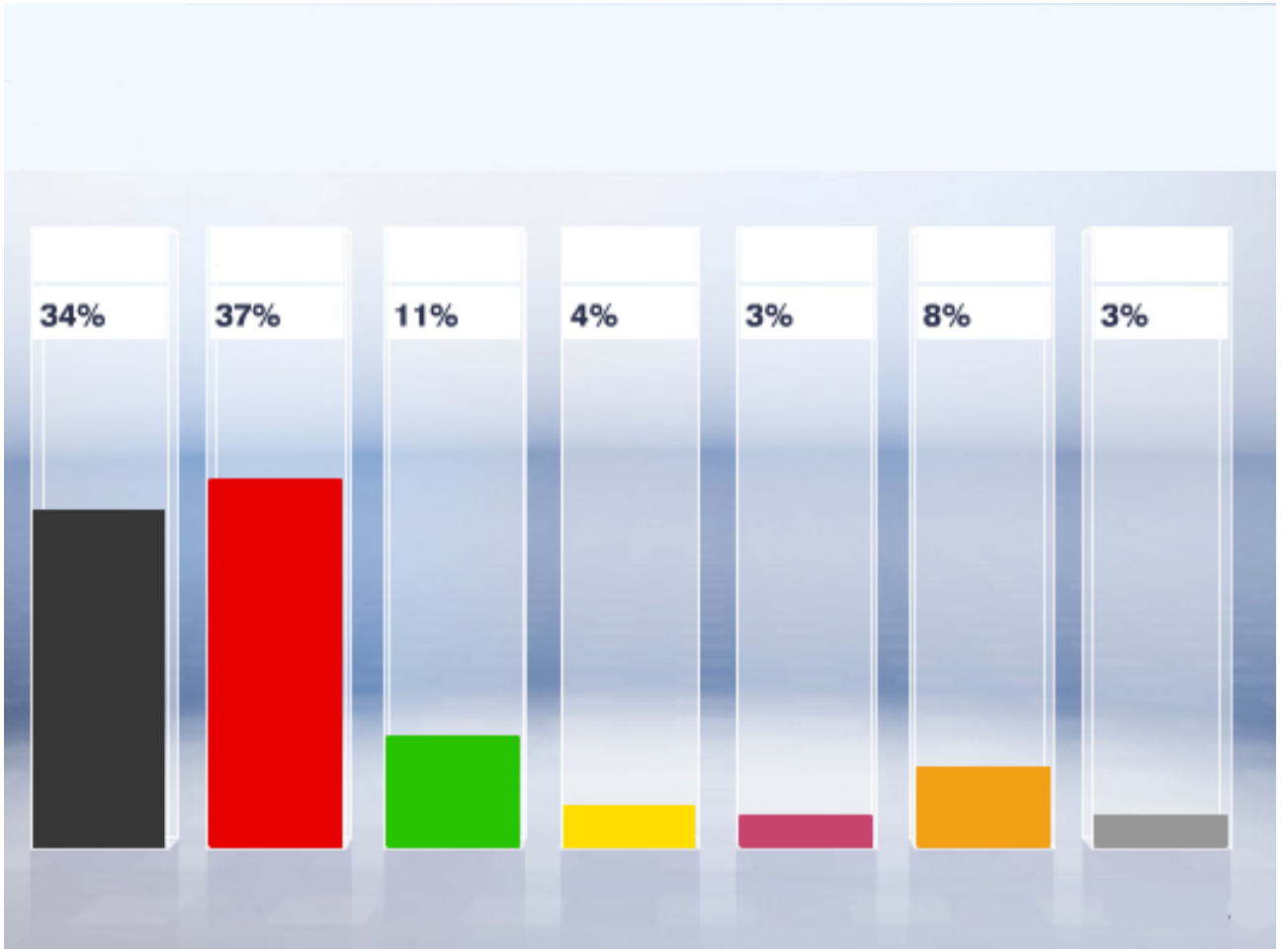
Problemstellung: Visualisieren

- Was soll dargestellt werden?
  - Messwerte?
  - Verteilung von Messwerten?
  - Komplexe Messhierarchien?
  - Reduktion auf einfache Kenngrößen?
  - Statistische Kennwerte?

Problemstellung: Visualisieren

- Als Grafik?

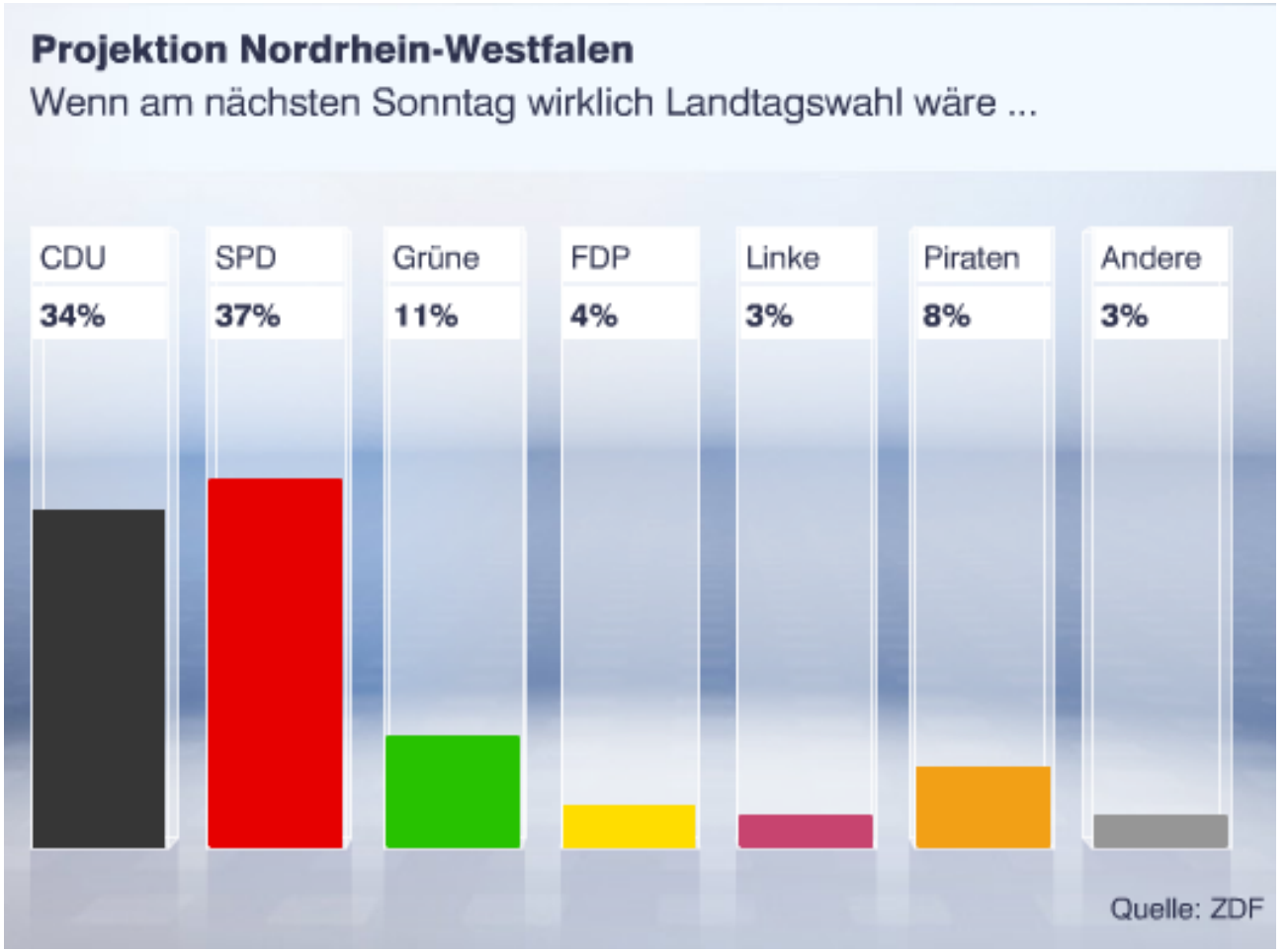
Besser,
aber
was ist
das?



Problemstellung: Visualisieren

- Als Grafik?

Was ist das? Eine Wahl-projektion kurz vor der NRW-Wahl im Mai 2012. Quelle ZDF



Problemstellung: Visualisieren

- Was genau soll dargestellt und veranschaulicht werden?
  - Messwerte einer Anwendung?
  - Messwerte von verteilten Anwendungen?
  - Messwerte einer Operation oder einer Methode einer Klasse?
  - Laufzeit eines Threads, mehrerer Threads?

Problemstellung: Visualisieren

- Was genau soll dargestellt und veranschaulicht werden?
  - Messwerte einer Anwendung?
  - Messwerte von verteilten Anwendungen?
  - Messwerte einer Operation oder einer Methode einer Klasse?
  - Laufzeit eines Threads, mehrerer Threads?
- Konzepte werden benötigt

Problemstellung: Visualisieren

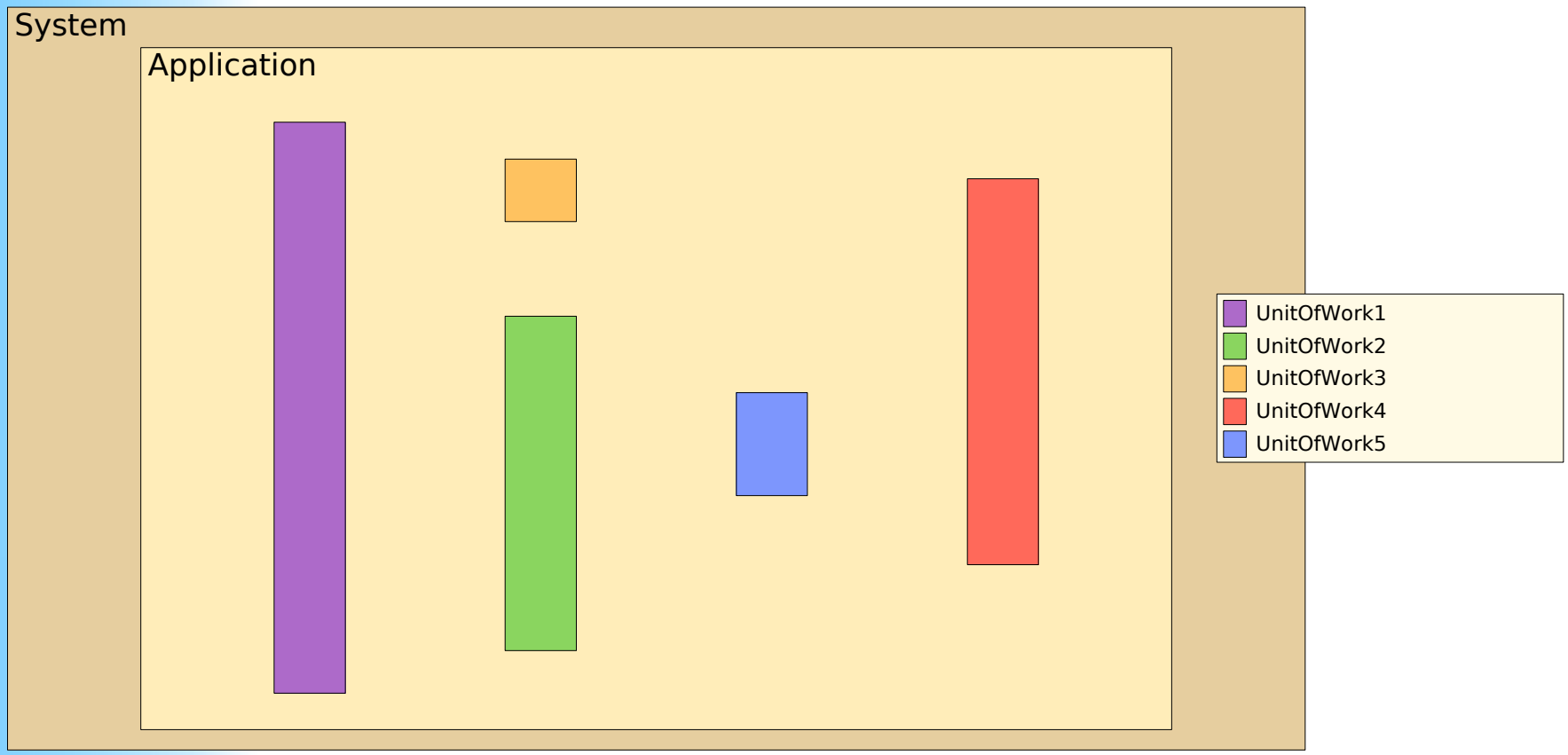
- Welche Konzepte? Applikation (Prozess)

Problemstellung: Visualisieren

- Welche Konzepte? System

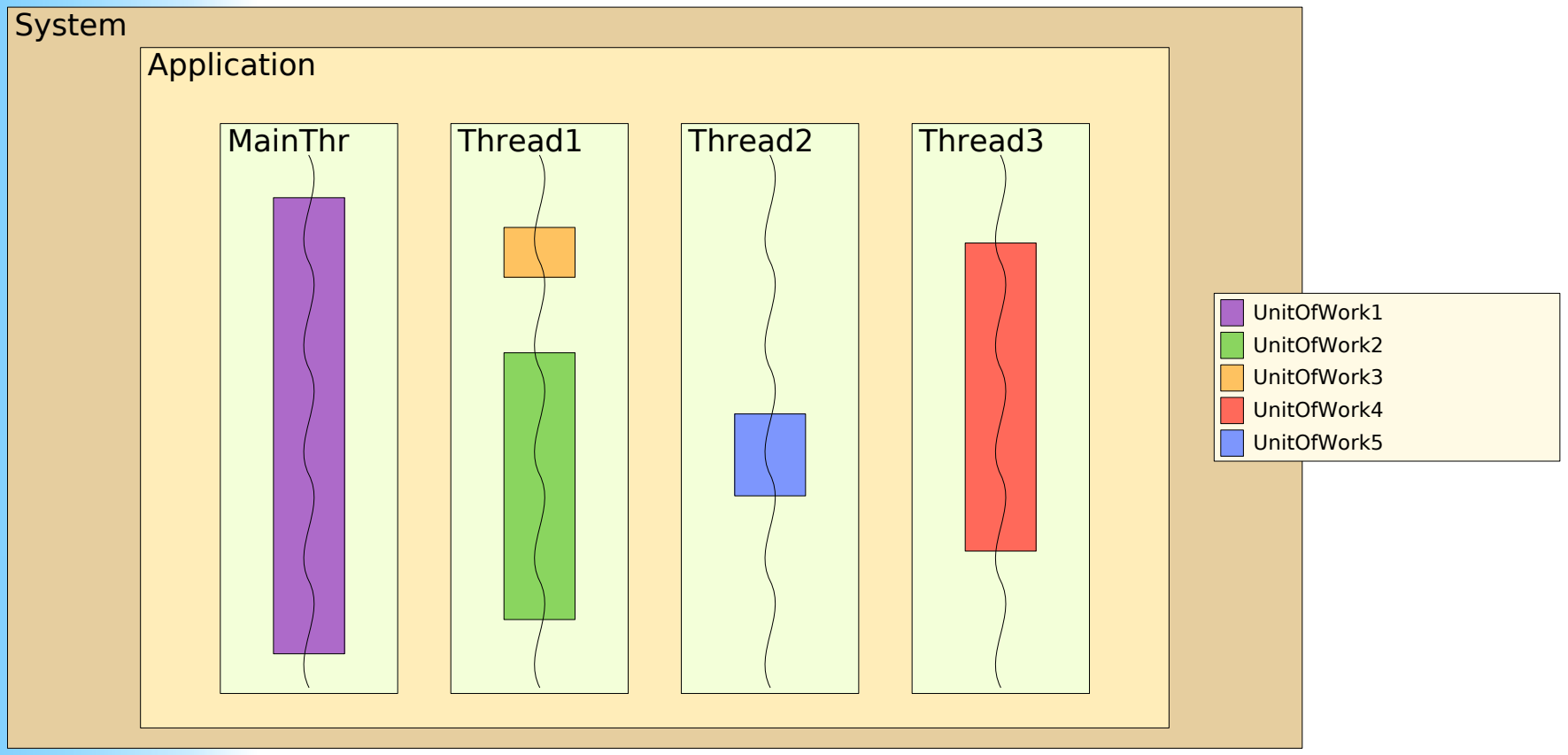
Problemstellung: Visualisieren

- Welche Konzepte? Arbeitseinheiten



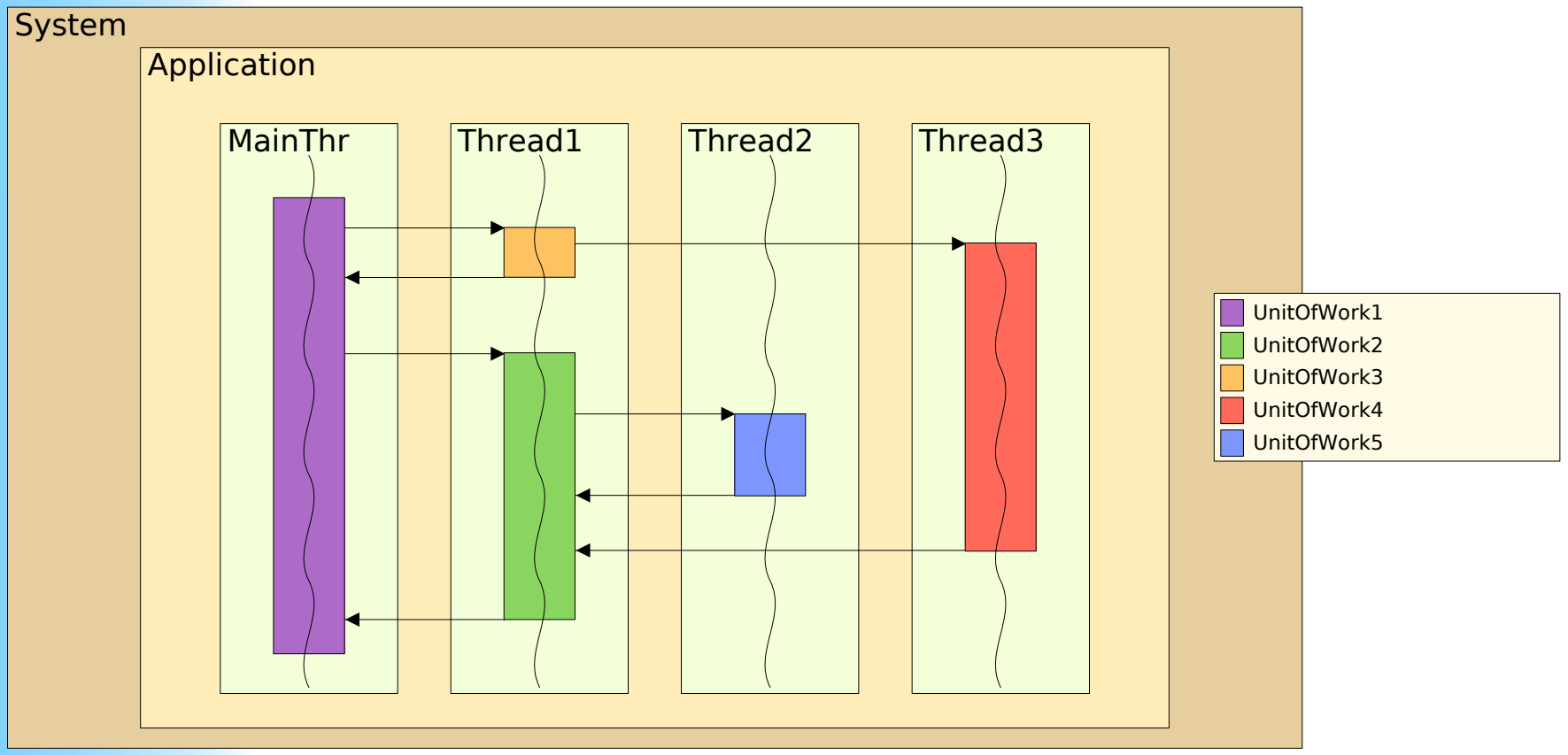
Problemstellung: Visualisieren

- Welche Konzepte? Threads



Problemstellung: Visualisieren

- Welche Konzepte? Aufrufbeziehungen



Problemstellung: Visualisieren

- Was soll dargestellt werden?
  - Messwerte?
  - Verteilung von Messwerten?
  - Komplexe Messhierarchien?
  - Reduktion auf einfache Kenngrößen?
  - Statistische Kennwerte?

Problemstellung: Visualisieren

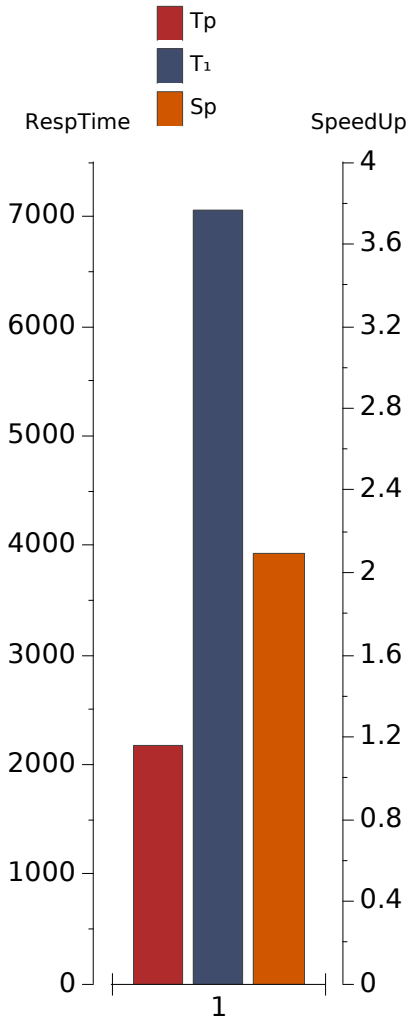
- Parallelität?
  - Einfache Kenngröße: „SpeedUp“ oder auch Amdahlsches Gesetz
  - Vereinfacht: Gibt den Beschleunigungs-faktor an, der durch Parallelisierung von Arbeitseinheiten im Vergleich zur sequentiellen Ausführung erreicht werden kann.

Problemstellung: Visualisieren

- Parallelität?
  SpeedUp:

$$S_p = \frac{T_1}{T_p} = \frac{T_1}{\left(T_1 * \left(\left(1-f\right) + \frac{f}{p}\right)\right)}$$

S_p: SpeedUp
T_1: Antwortzeit mit einer CPU
T_p: Antwortzeit mit p CPUs
f : Prozentualer Anteil parallelisierbarer Arbeitseinheiten



Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

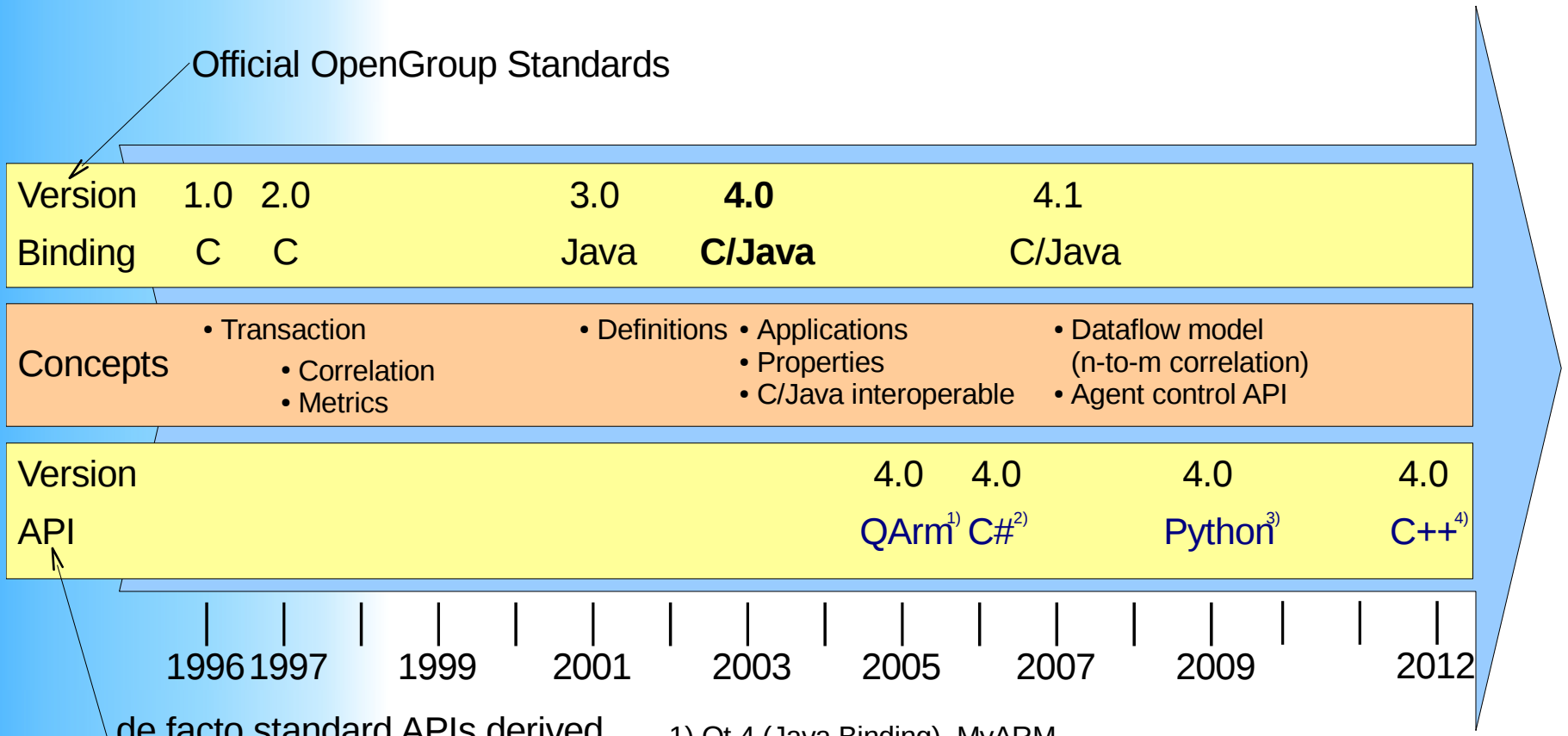
Instrumentierung der Anwendung

- Anwendung kennt und implementiert die geforderte Funktionalität
- Geeignete Arbeitseinheit (AE) zum Messen finden und benennen
- Anwendung instrumentieren (ARM):
  - Start- und Endzeitpunkt der Arbeitseinheit festhalten
  - Kein Debug-Werkzeug sondern Bestandteil der Anwendung (Auto ohne Tacho?)

Instrumentierung: Was ist ARM?

- Application Response Measurement (ARM): ein von der OpenGroup angenommener Industriestandard
- definiert Konzepte und Schnittstellen (API) für die Messung der Antwortzeit von Arbeitseinheiten
- API für versch. Programmiersprachen: C und Java (C++, C# und Python)
- Korrelation verteilter Arbeitseinheiten

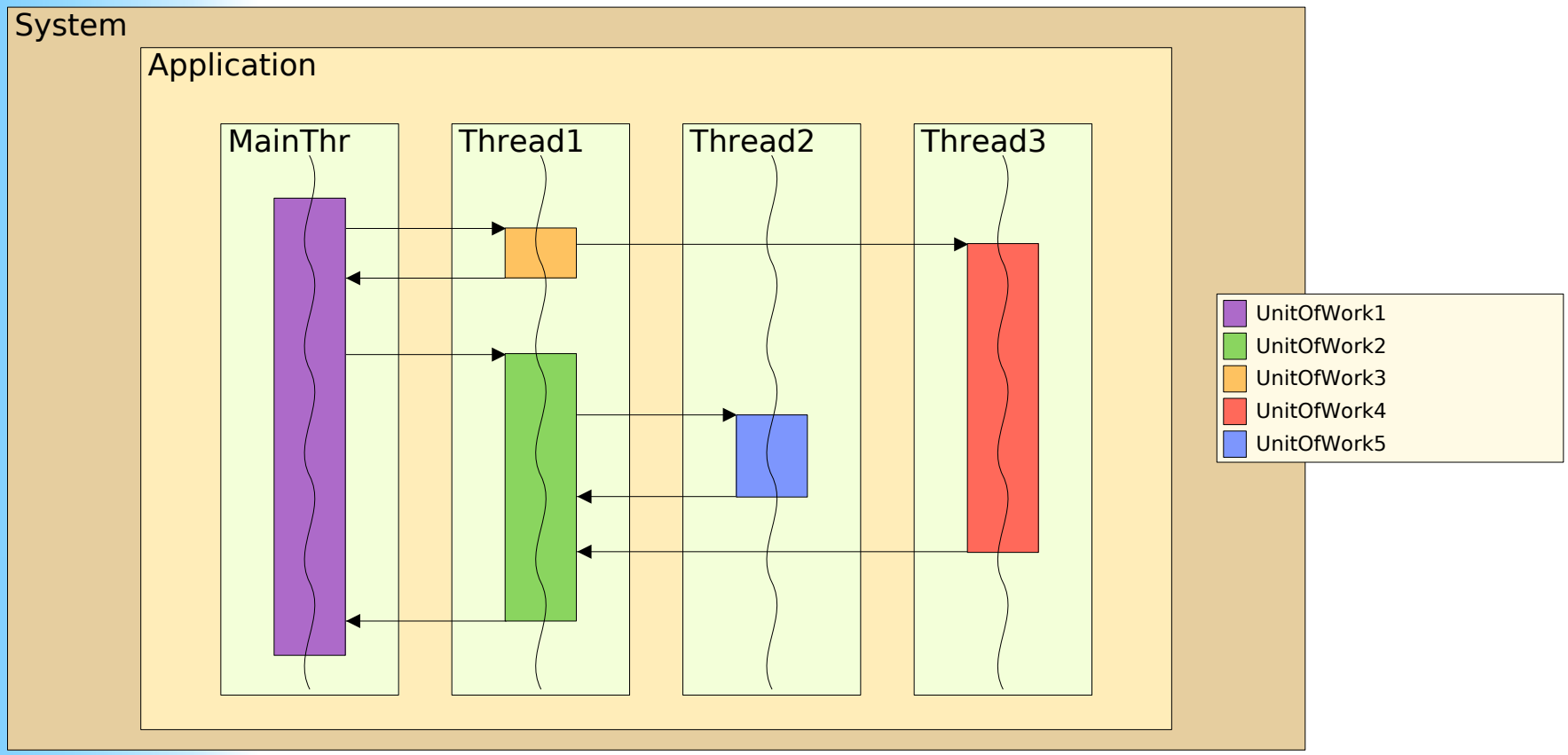
Instrumentierung: ARM-Historie



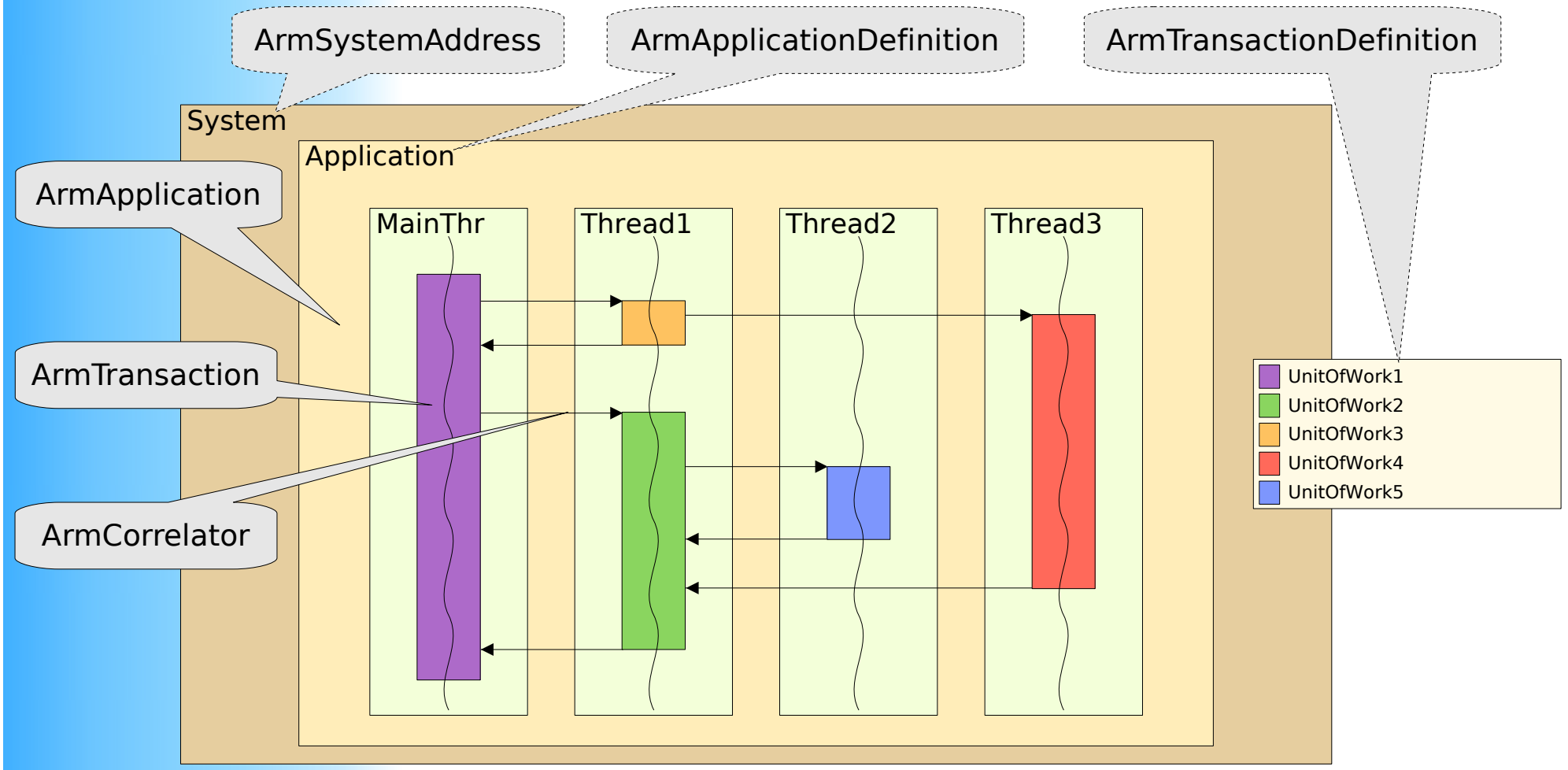
1) Qt 4 (Java Binding), MyARM
2) C# .NET (Java Binding) Eclipse/MyARM
3) Python (C Binding) arm4.org/Fedora/MyARM
4) C++ (Java Binding), MyARM

Instrumentierung: ARM-Konzepte

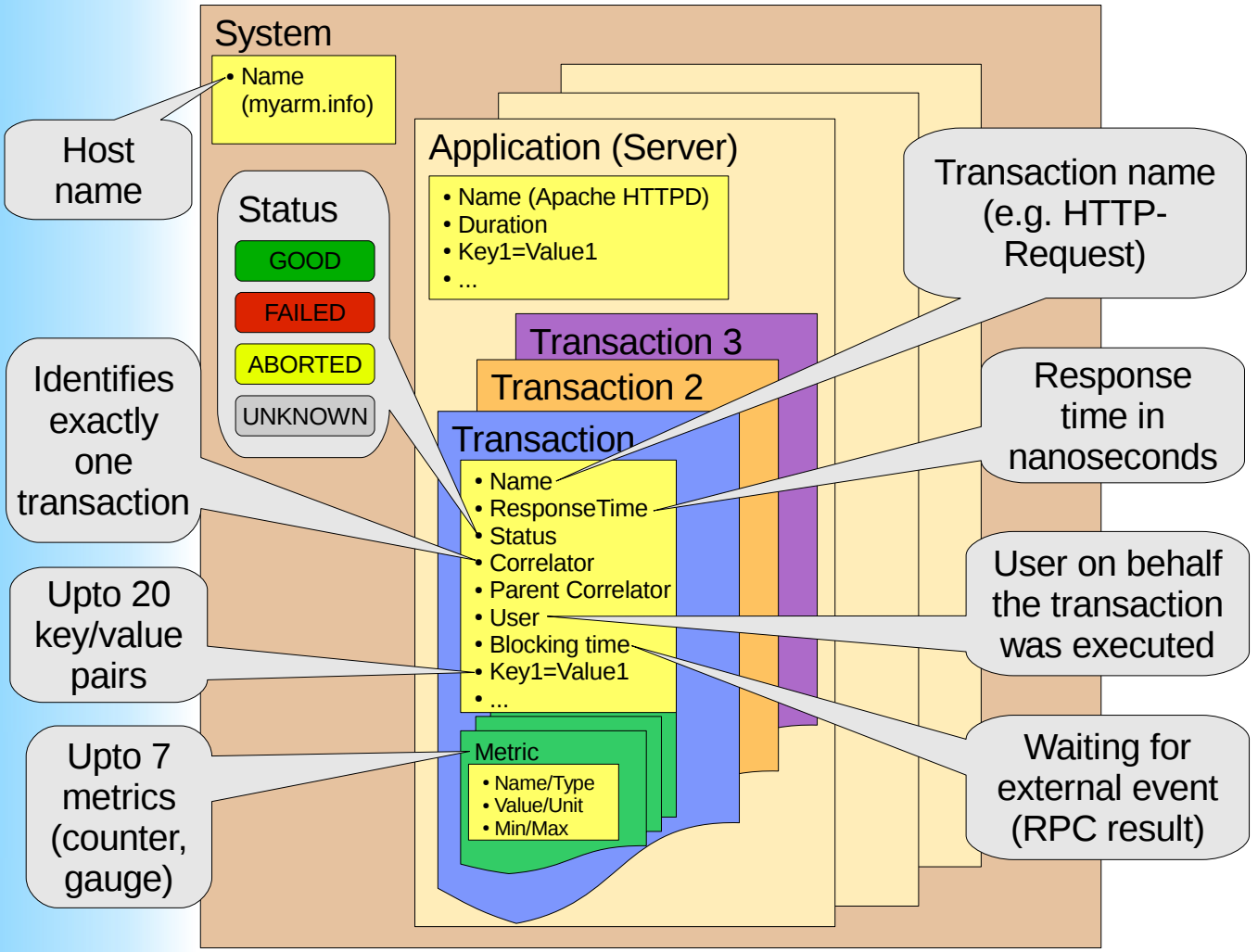
- Benötigte Konzepte:



Instrumentierung: ARM-Konzepte



Instrumentierung: ARM-Konzepte (Details)



Instrumentierung: C++ Beispiel

```
#include <ArmApplicationDefinition>
…
using namespace arm4;
…
ArmApplicationDefinition appDef("HelloWorldApp");
ArmTransactionDefinition tranDef(appDef, "HelloWorld");
```

Instrumentierung: C++ Beispiel

```
#include <ArmApplicationDefinition>
…
using namespace arm4;
…
ArmApplicationDefinition appDef("HelloWorldApp");
ArmTransactionDefinition tranDef(appDef, "HelloWorld");

ArmApplication app(appDef);
ArmTransaction tran(app, tranDef);
```

Instrumentierung: C++ Beispiel

```
#include <ArmApplicationDefinition>
…
using namespace arm4;
…
ArmApplicationDefinition appDef("HelloWorldApp");
ArmTransactionDefinition tranDef(appDef, "HelloWorld");

ArmApplication app(appDef);
ArmTransaction tran(app, tranDef);

tran.start();
std::cout << "Hello world" << std::endl;
tran.stop(ArmConstants::STATUS_GOOD);
```

Instrumentierung: C++ Beispiel

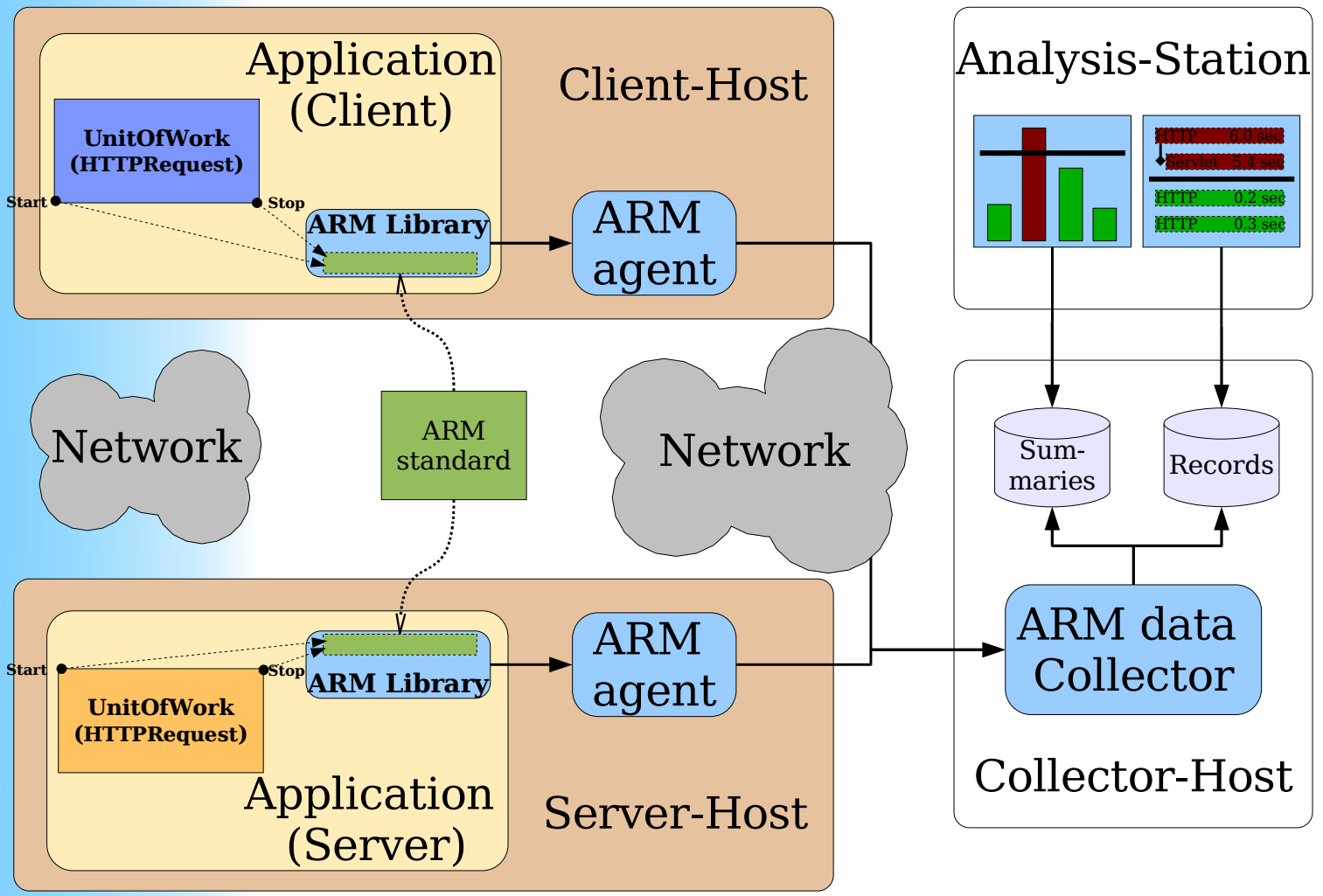
```
#include <ArmApplicationDefinition>
…
using namespace arm4;
…
ArmApplicationDefinition appDef("HelloWorldApp");
ArmTransactionDefinition tranDef(appDef, "HelloWorld");

ArmApplication app(appDef);
ArmTransaction tran(app, tranDef);

tran.start();
std::cout << "Hello world" << std::endl;
tran.stop(ArmConstants::STATUS_GOOD);

app.end();
appDef.destroy();
```

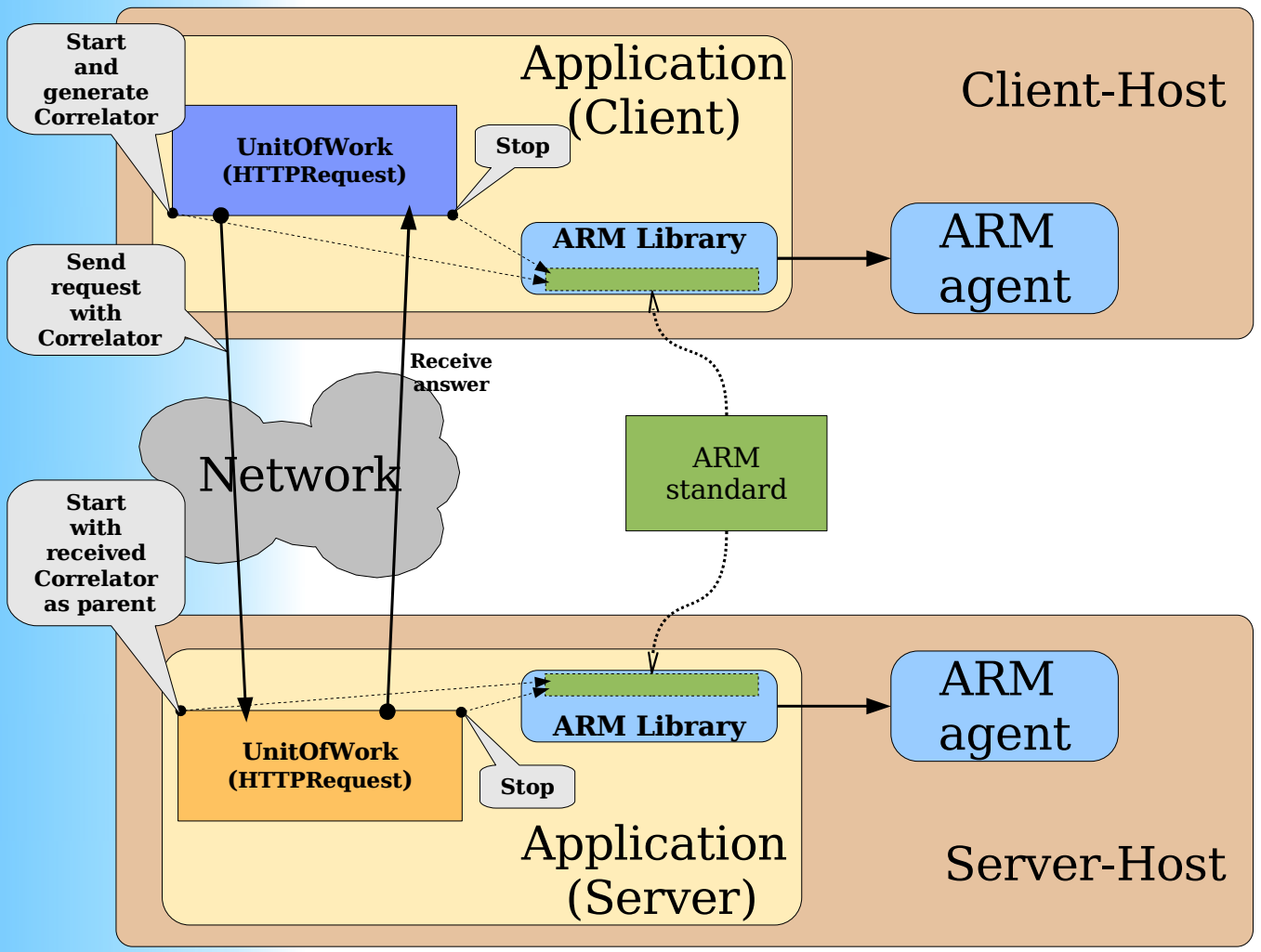
Instrumentierung: ARM Architektur



Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

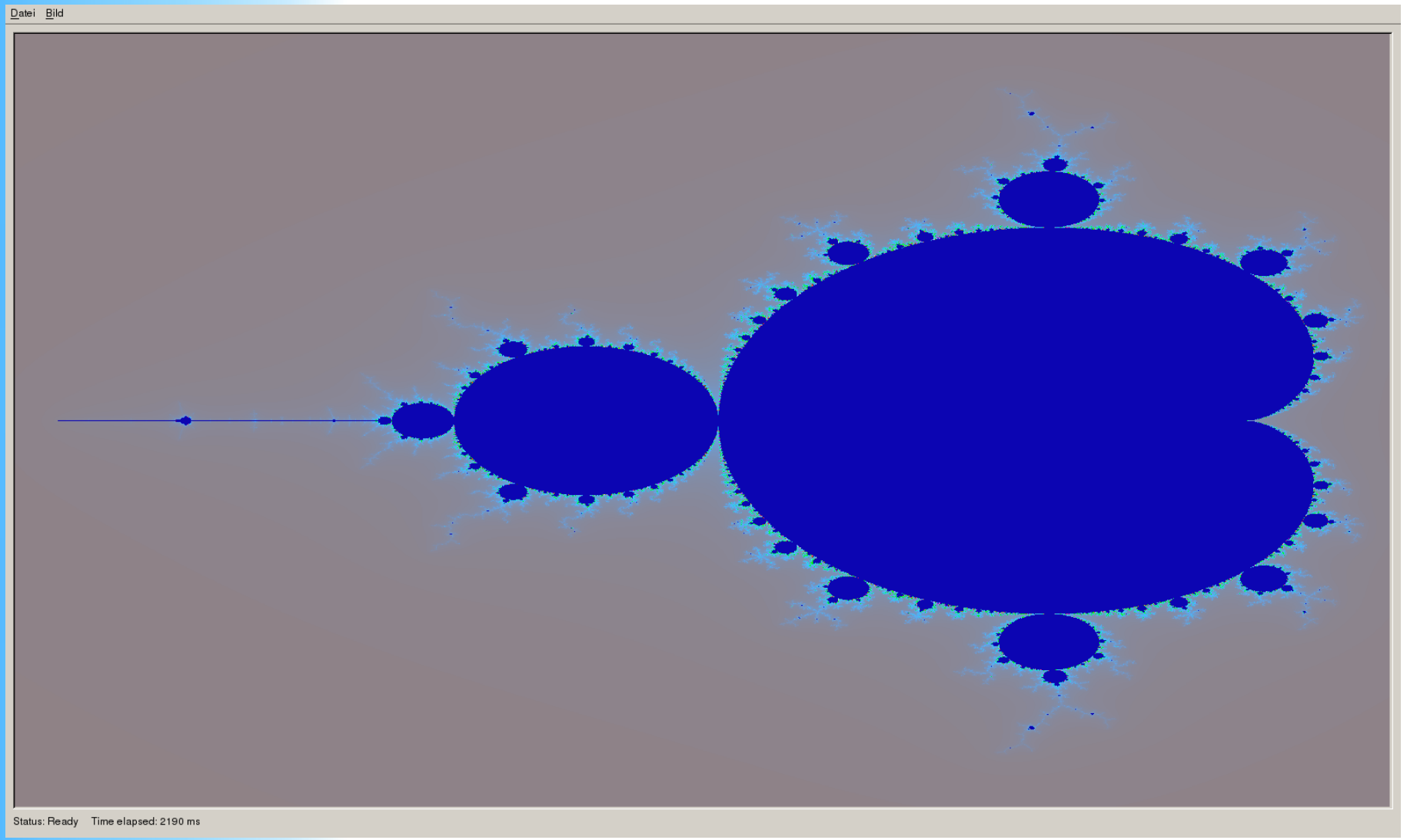
Instrumentierung: ARM Korrelation



Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

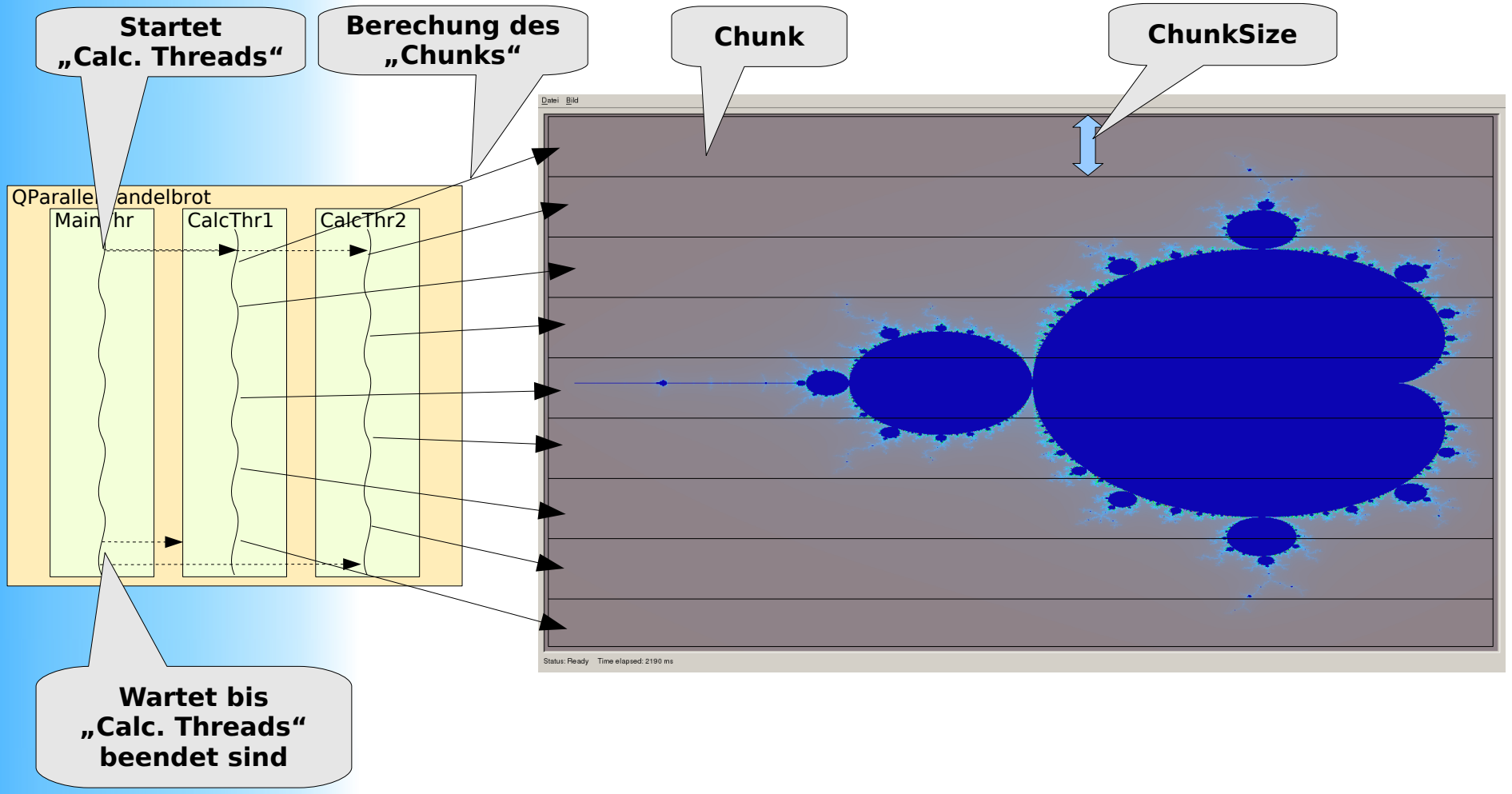
Parallelität: Beispiel Mandelbrot



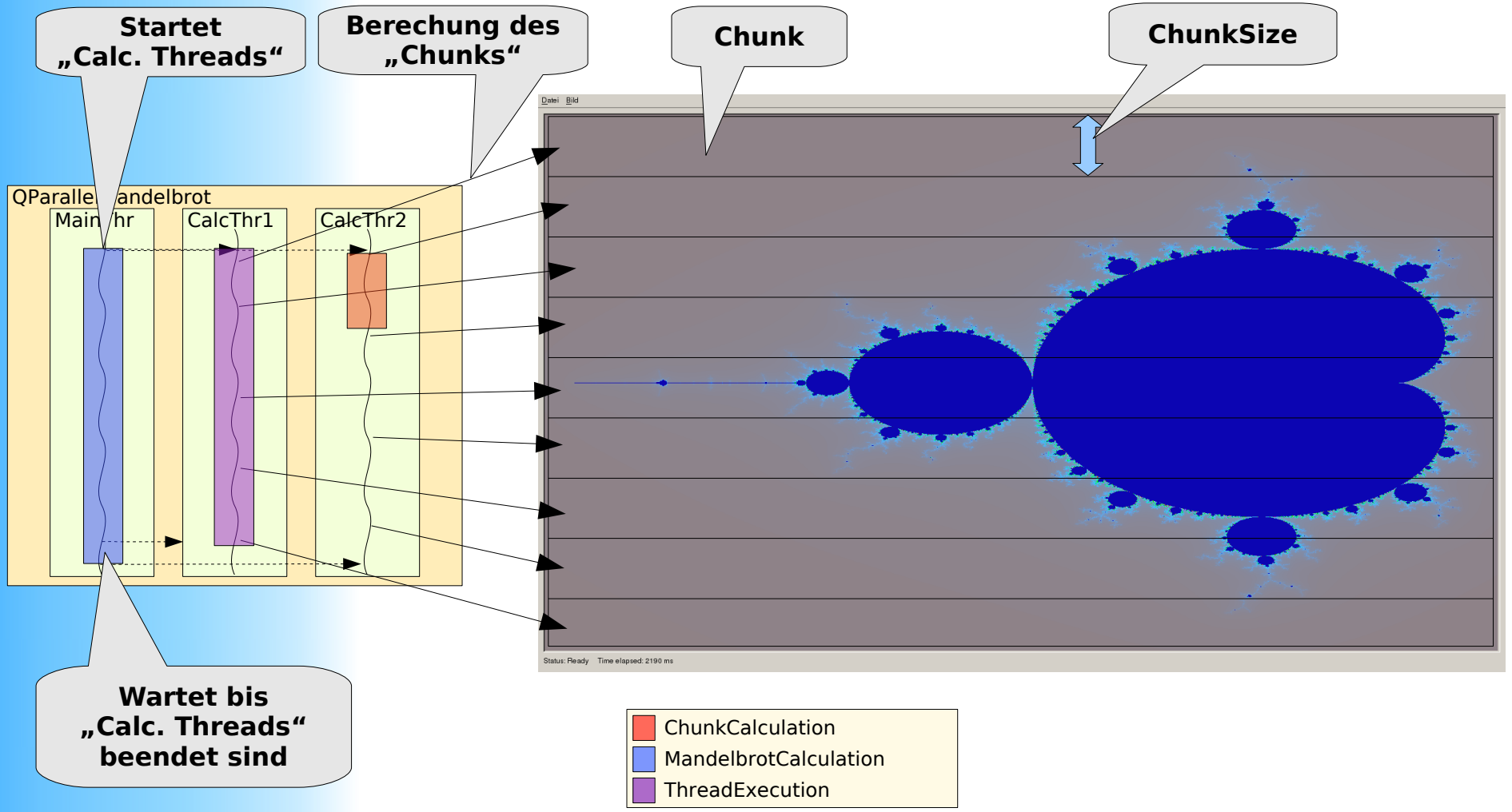
Parallelität: Beispiel Mandelbrot

- Qt 4 basierte QParallelMandelbrot Anwendung von Dr. Matthias Nagorni (auf heise developer)
- Parallelisierung durch Multithreading (Multicore Ausnutzung)
- Instrumentierung mittels QArm 4
  - An Qt 4 angepasstes ARM 4.0 API

Parallelität: Mandelbrot Anwendungsstruktur



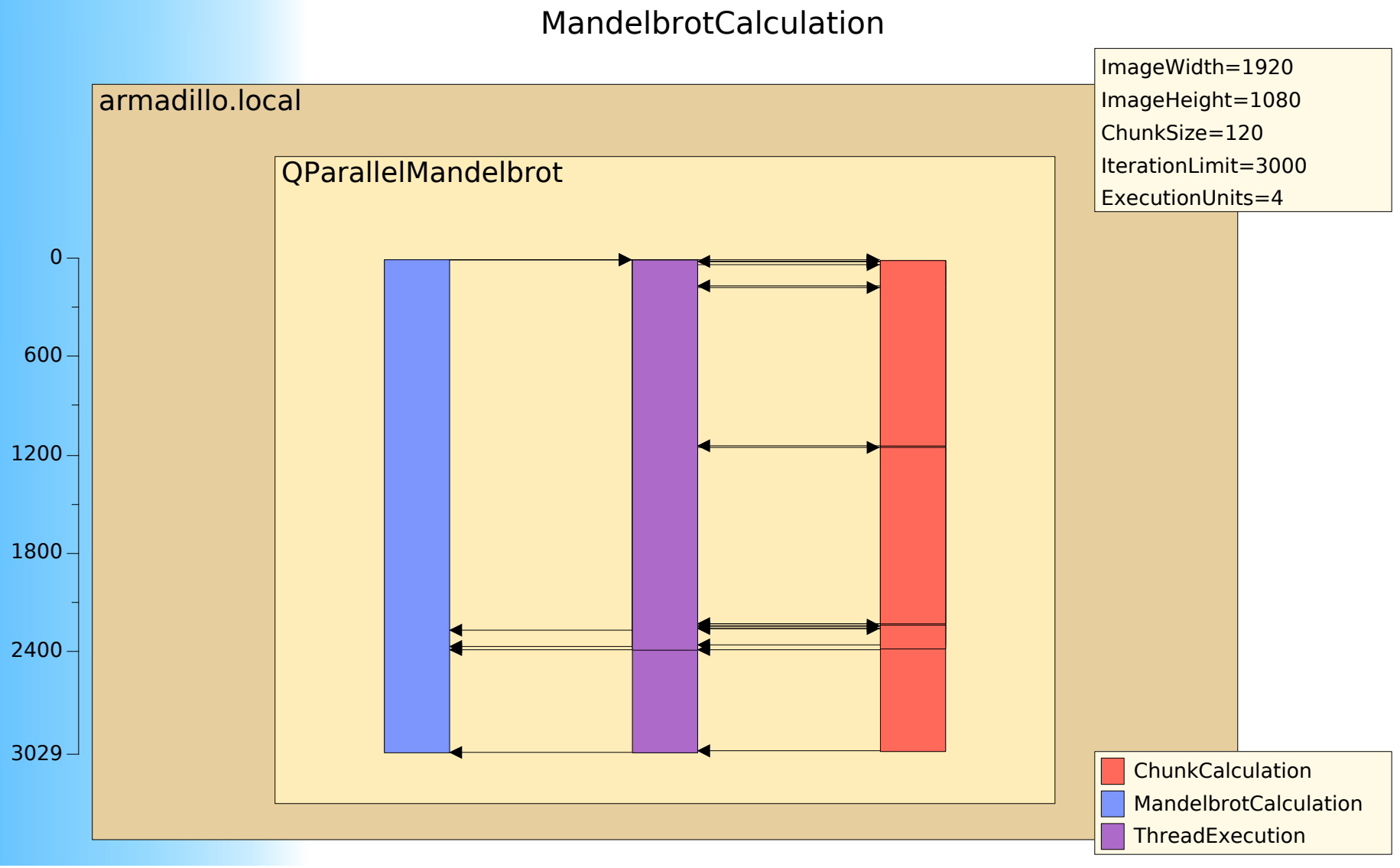
Parallelität Mandelbrot: Instrumentierung



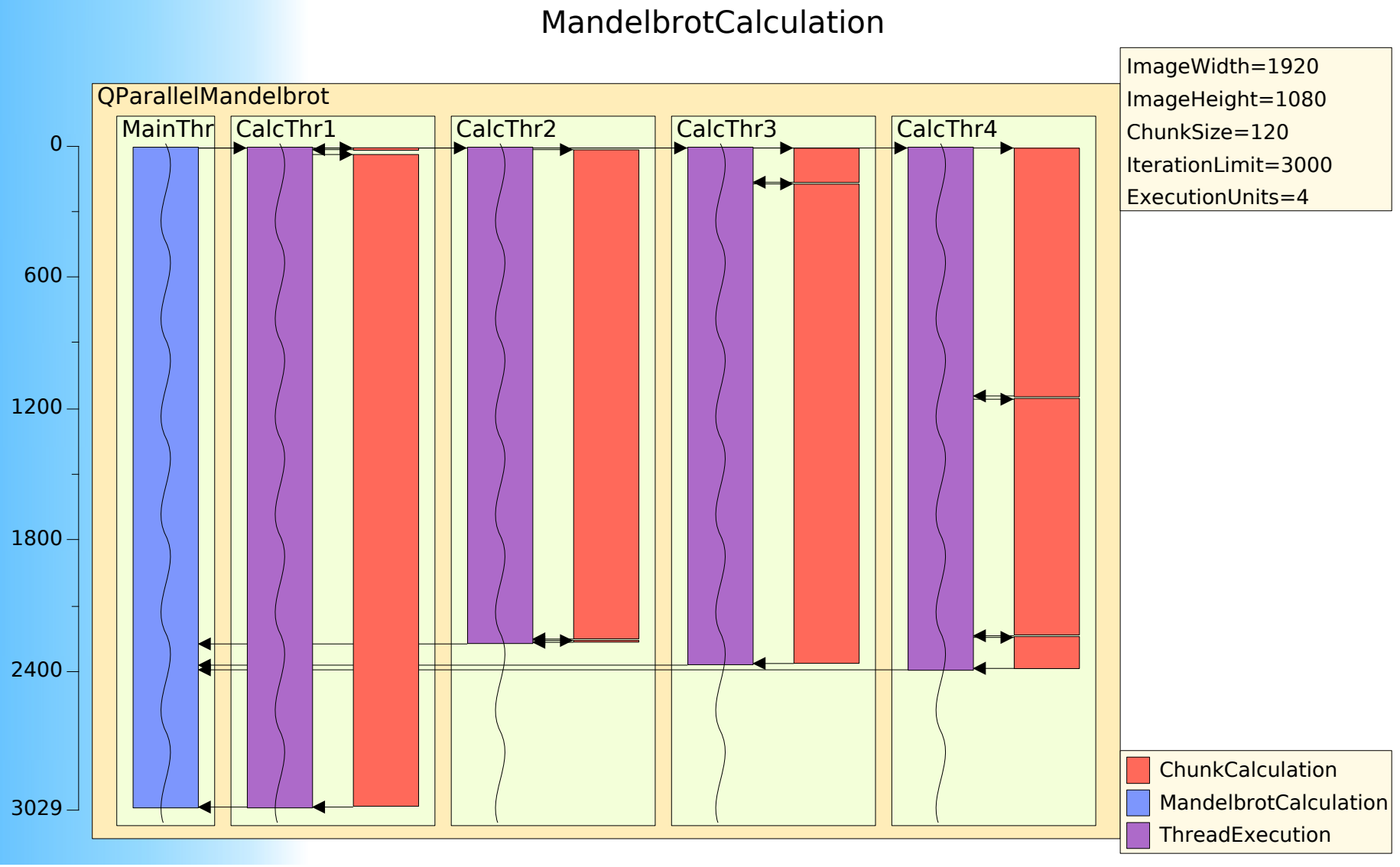
Inhalt

- Einleitung
  - Messen
  - Visualisieren
- Instrumentierung
  - ARM Standard und Architektur
  - Korrelation von Messwerten
- Parallelität
  - am Beispiel von Mandelbrot
  - Messwerte visualisieren und einordnen

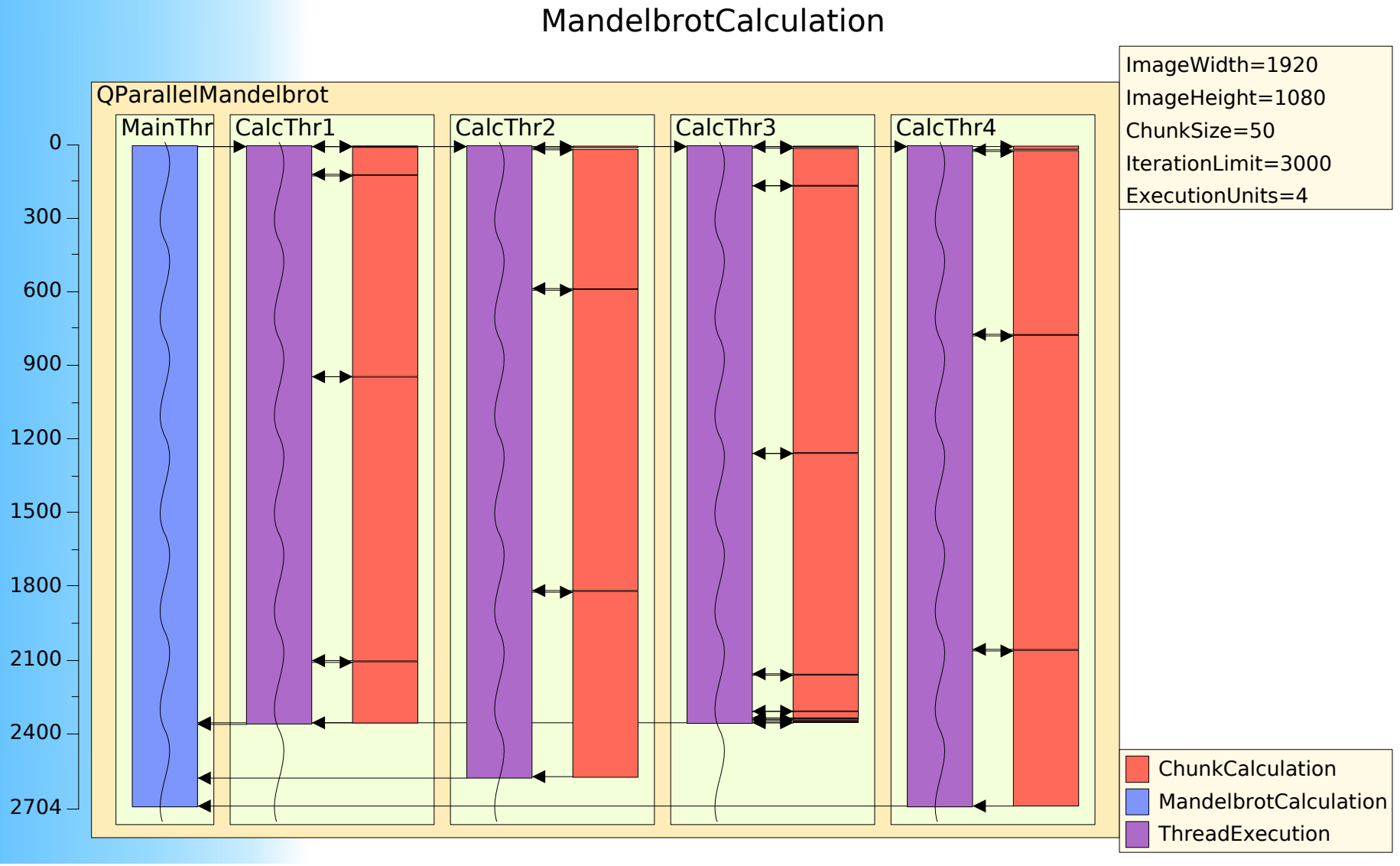
Mandelbrot – Messung (1)



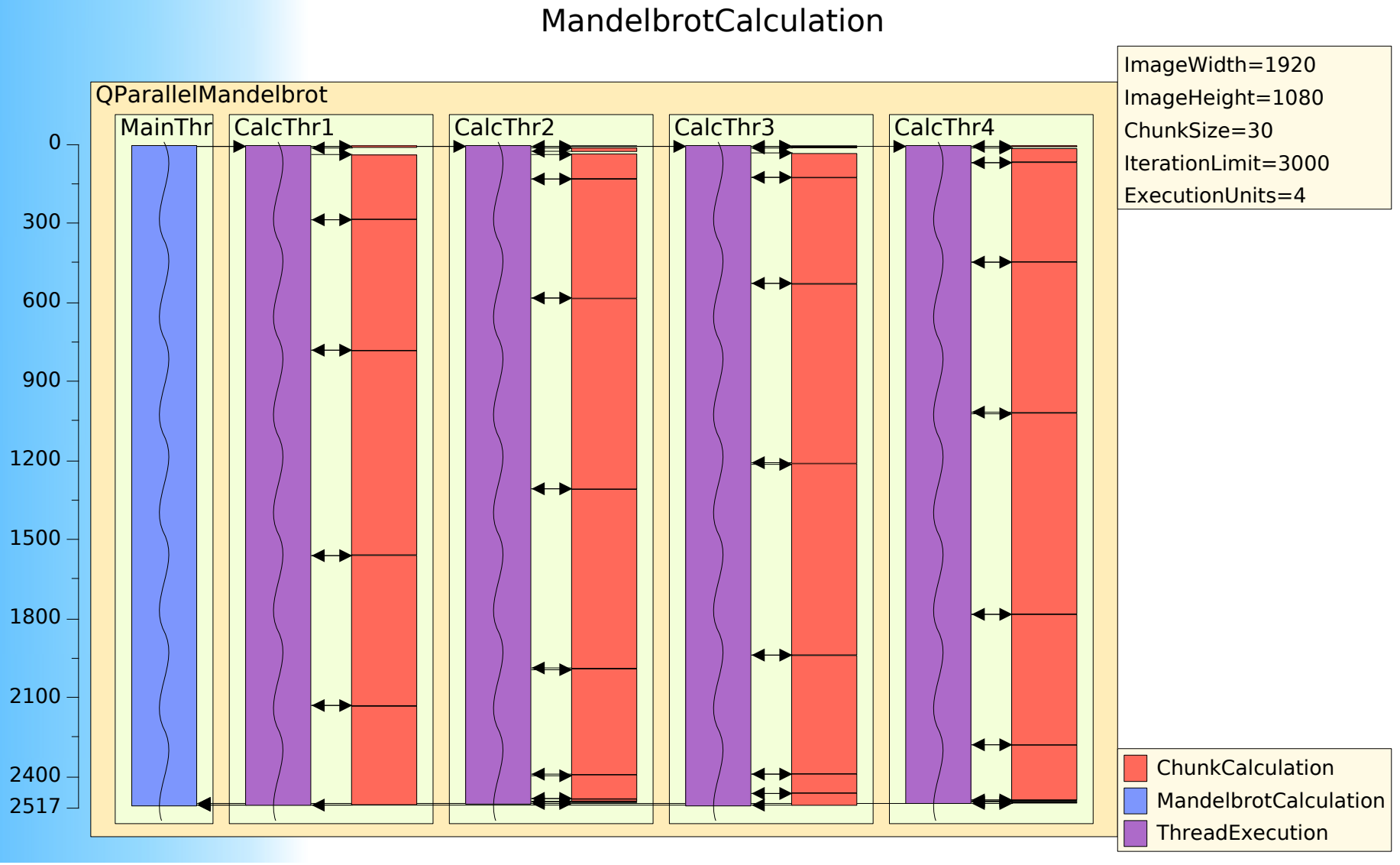
Mandelbrot – Messung (2)



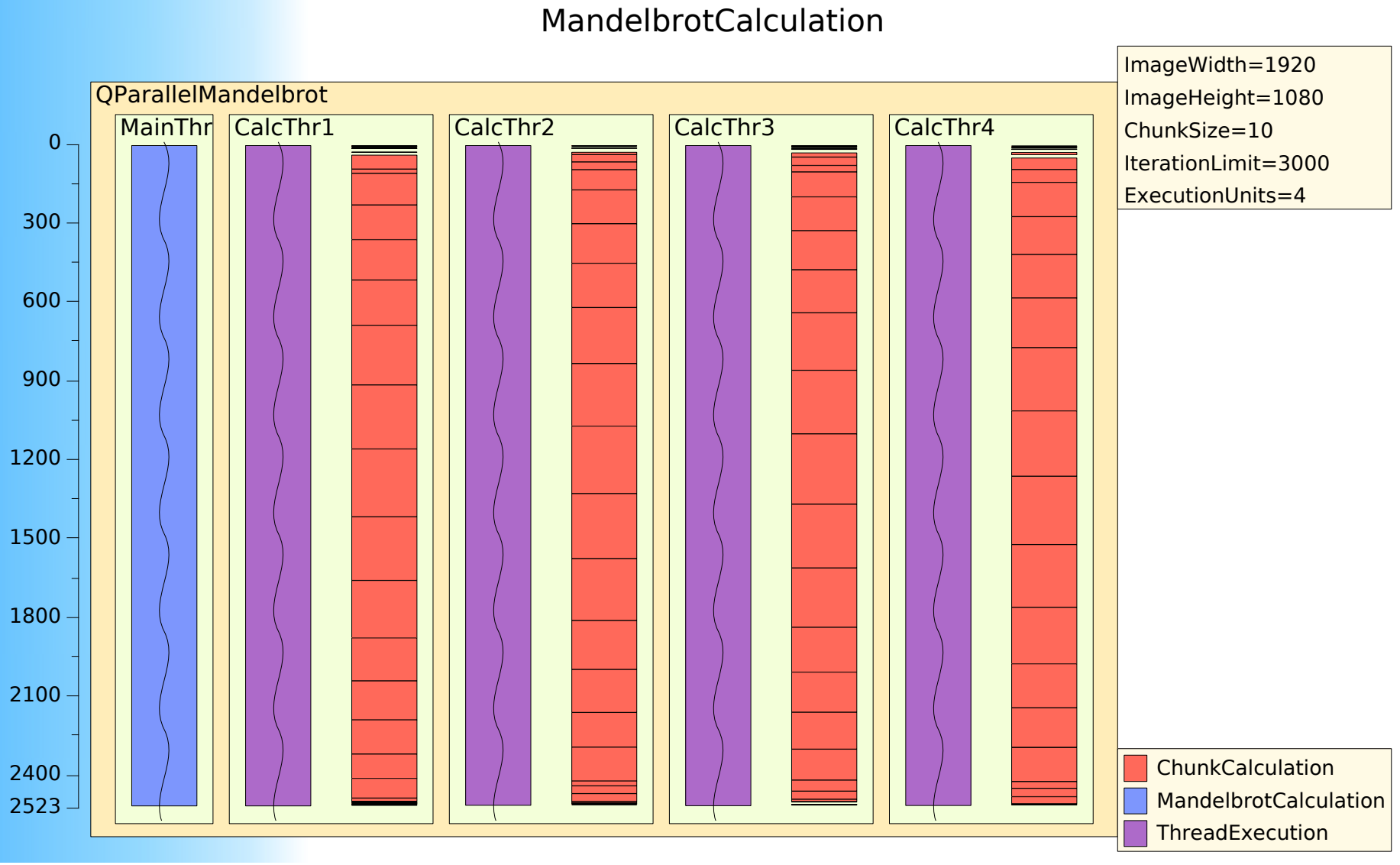
Mandelbrot – Messung (3)



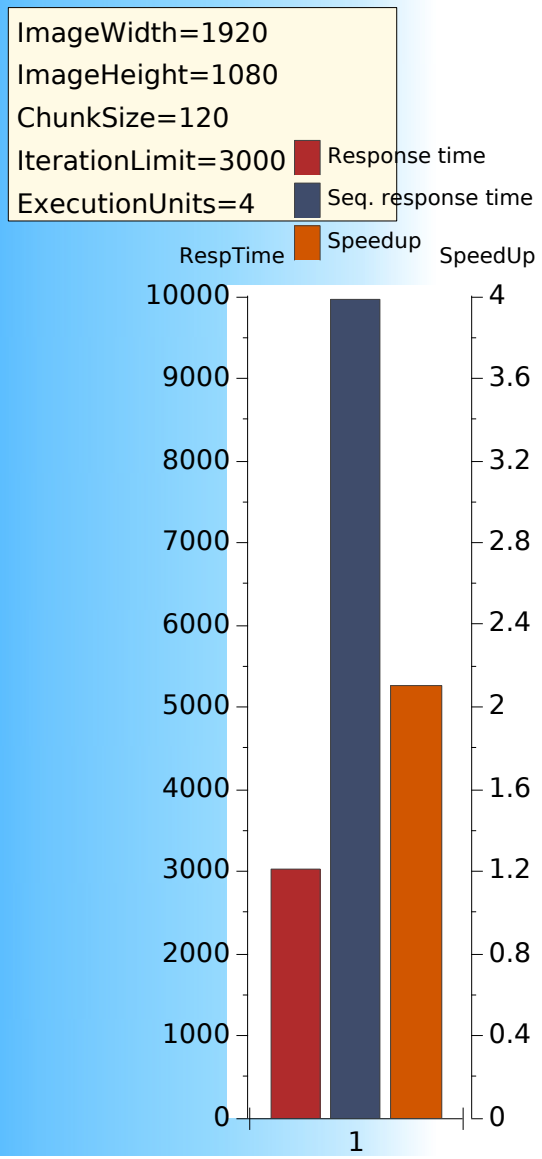
Mandelbrot – Messung (4)

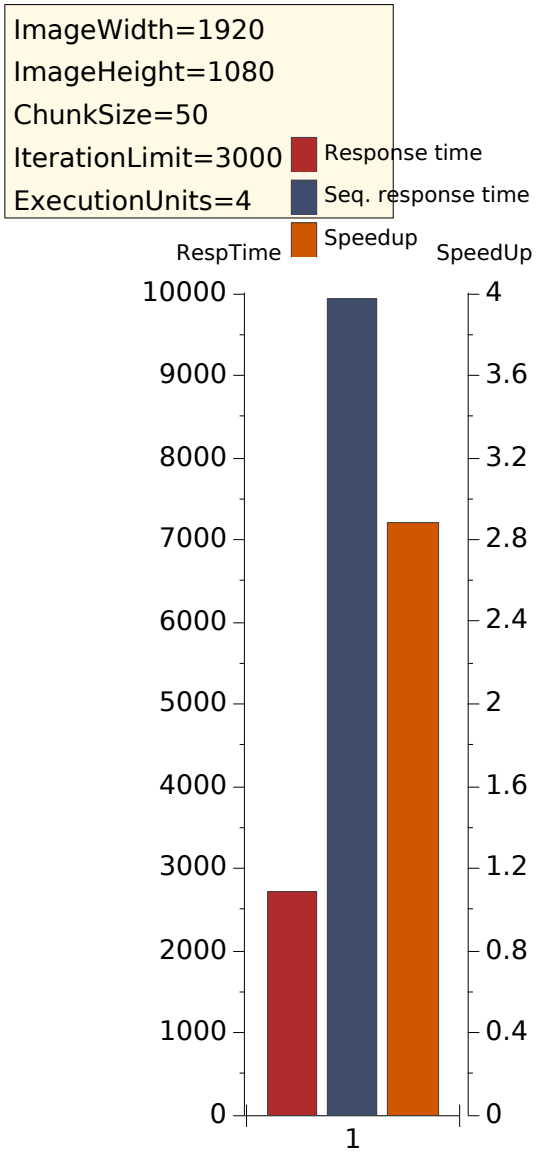


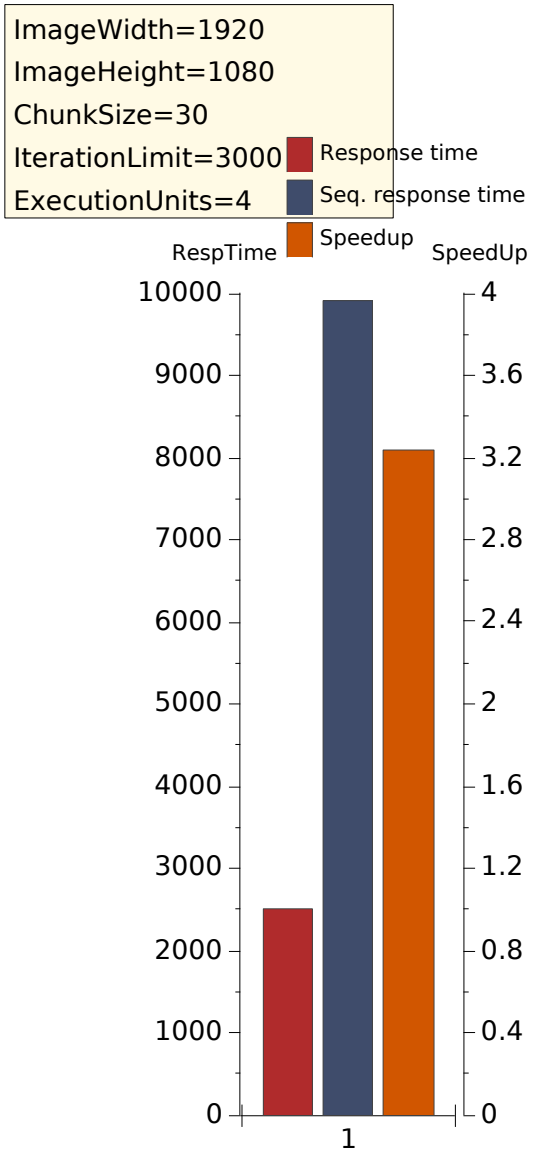
Mandelbrot – Messung (5)



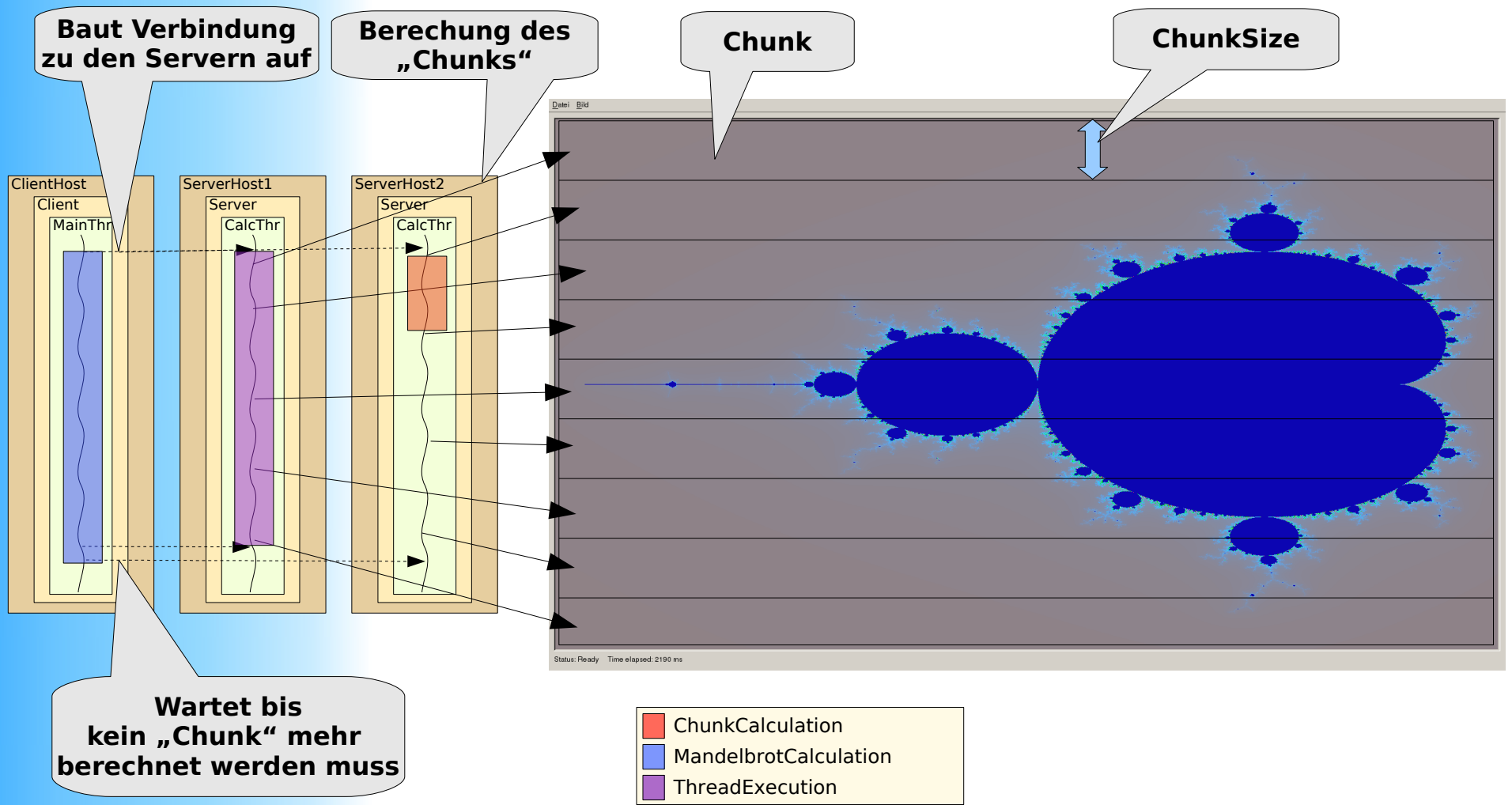
Mandelbrot: Messung (6)



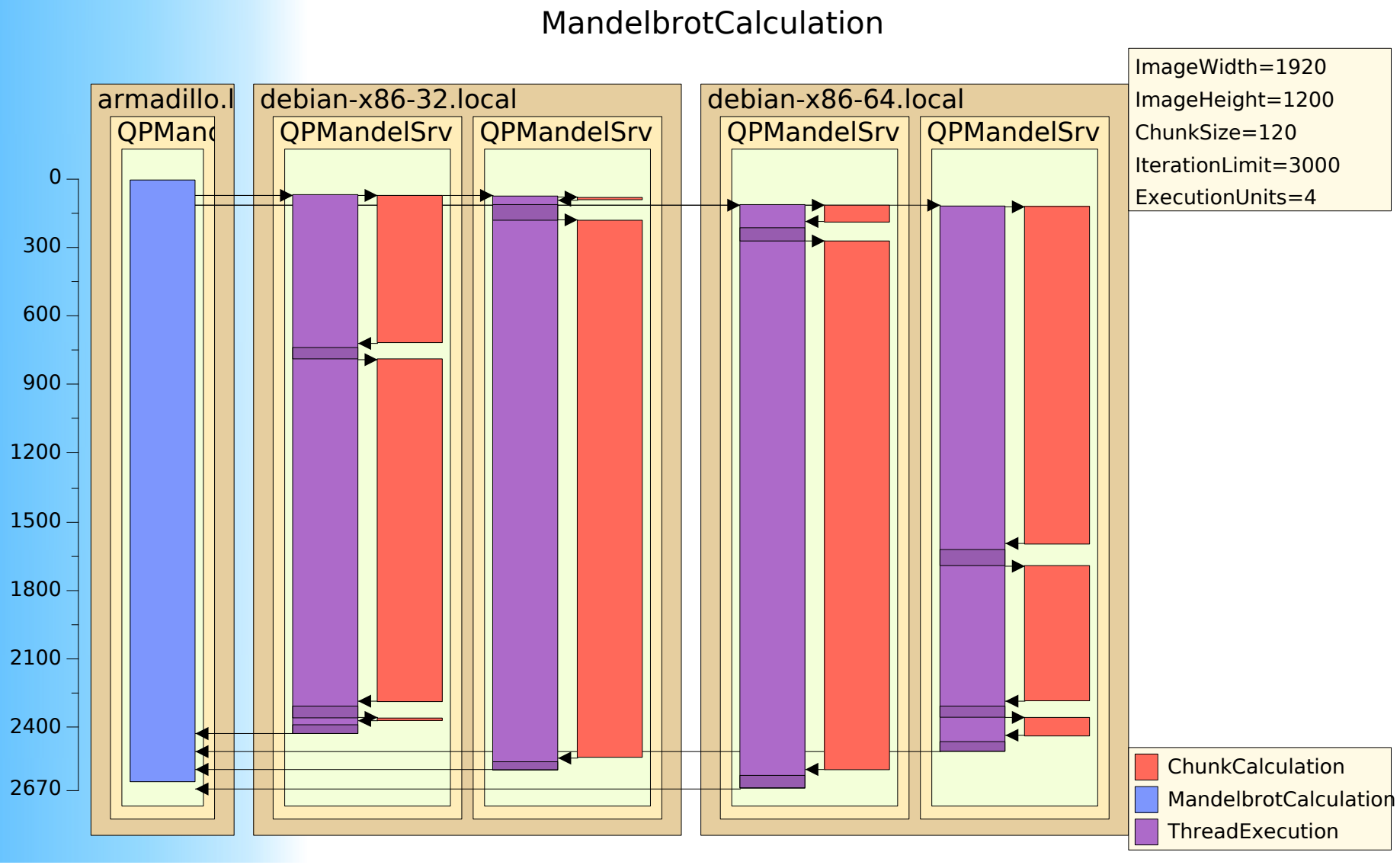




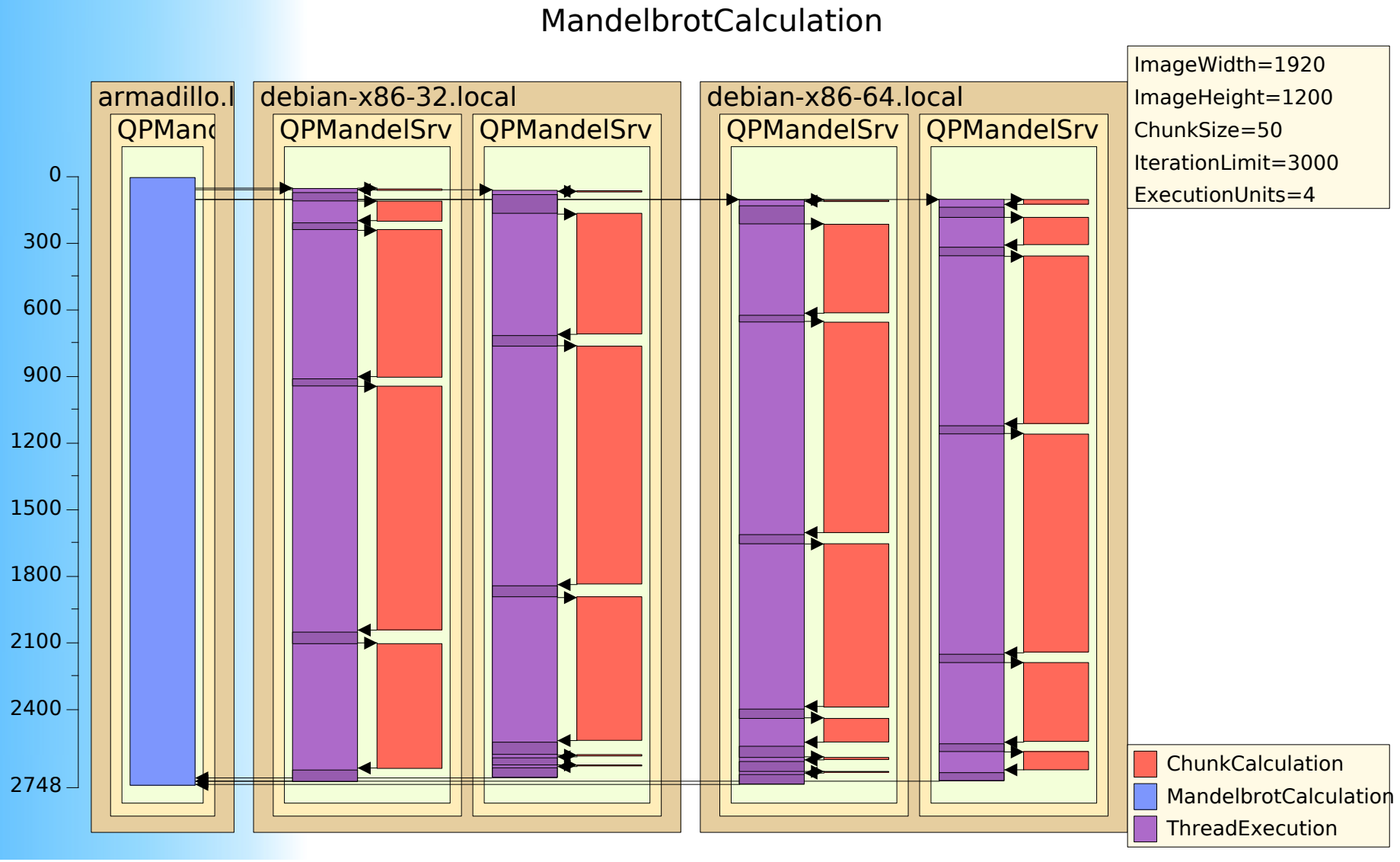
Mandelbrot: Client/Server (1)



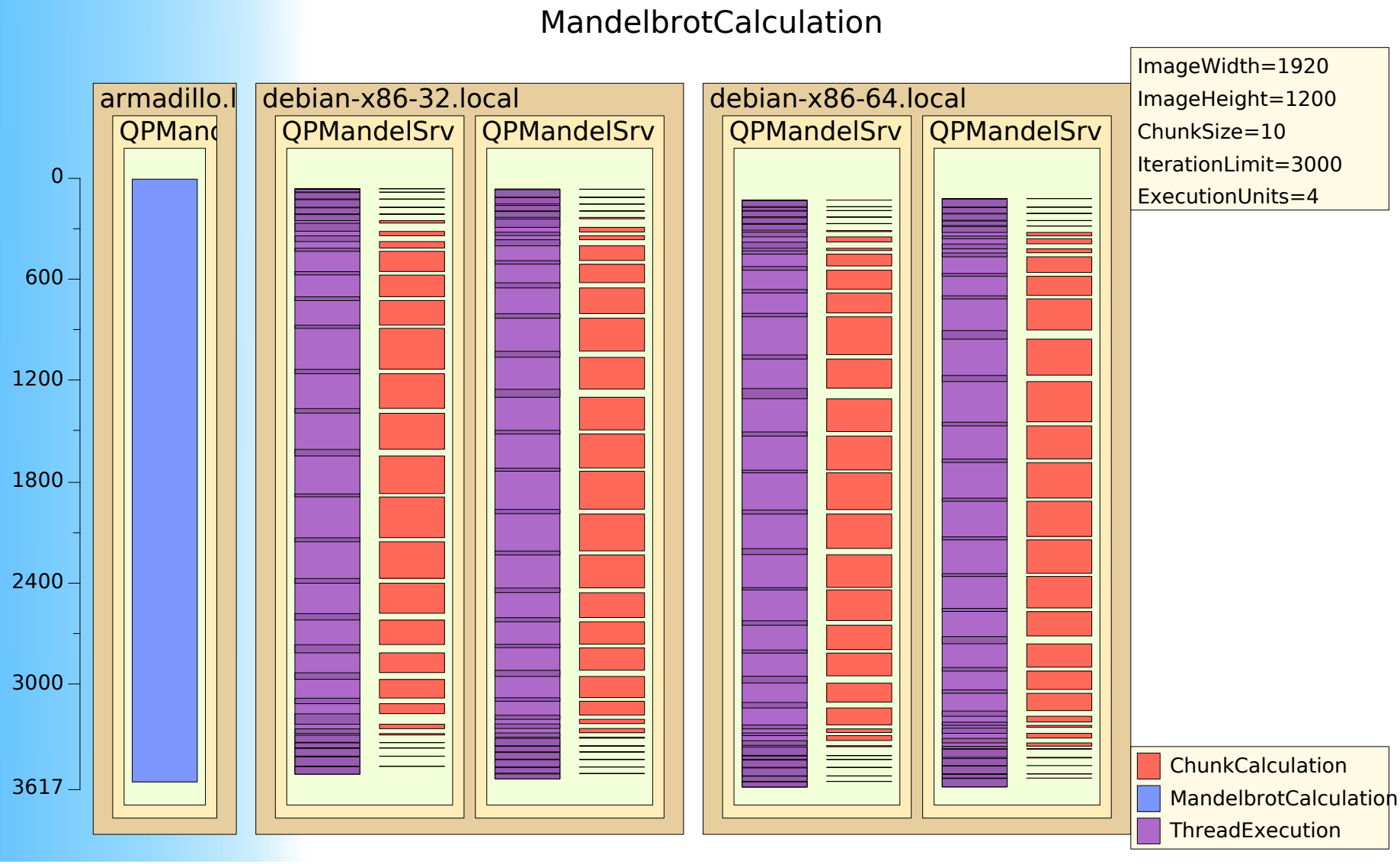
Mandelbrot – Client/Server (2)



Mandelbrot – Client/Server (3)



Mandelbrot – Client/Server (4)



Messen und Visualisieren:

- Einblick in das Laufzeitverhalten der Anwendung
- Analyse und Optimierung von Algorithmen
- Kann Laufzeitmuster erkennbar machen
- Ermöglicht Reduktion auf einfache Kenngrößen (SpeedUp)

Weitere Einsatzgebiete

- Ermittlung von KPIs möglich
- Tracing für Fehleranalyse
- Statistische Aggregation
- Einblick in verteilte Anwendungen
- Ggf. Korrelation mit Log-Einträgen
- SLAs

Zusammenfassung

- Antwortzeit als Maß für (Performance-) Messungen
- Korrelation von Messwerten mittels ARM-Korrelatoren
- Visualisierung von Messwerten in komplexen und parallelen Umgebungen
- Analyse und Optimierung der Anwendung

Backup

Kennzahlen (MyARM)

- Call-Overhead für Java typisch 2,5 - 6 µs
- Call-Overhead für C typisch 1 - 5 µs
- Ca. 100 Byte je Messung serialisiert
- 12.000-15.000 Inserts je Sekunde in DB
- Ca. 200.000 Messungen/s in Applikation
- Implementierung entkoppelt von Applikation (Threads), Speicher-Backend
  - Keine Störung der Applikation

Sprachbindung (1)

C Schnittstelle

- `arm_register_application()`
- `arm_register_transaction()`
- `arm_register_metric()`

- `arm_start_application()`
- `arm_stop_application()`

- `arm_start_transaction()`
- `arm_stop_transaction()`

Java Schnittstelle

- `ArmApplicationDefinition`
- `ArmTransactionDefinition`
- `ArmMetricDefinition`

- `ArmTranFactory.newArmApplication()`
- `ArmApplication.end()`

- `ArmTransaction.start()`
- `ArmTransaction.stop()`

Sprachbindungen (2)

- Defacto Schnittstelle für C# und C++ (ARM 4.0)
- Skript-Sprachen Unterstützung durch Nutzung der C Schnittstelle für:
  - Python (ARM 4.0)
  - Shell-Skripte mittels armtime (ARM 4.0)
  - Perl (ARM 2.0)
  - PHP (ARM 2.0)
  - Ruby (möglich)