Visualisierung

Präsentation von Joshua Schimmelpfennig
Seminar Effiziente Programmierung
Universität Hamburg
Hamburg den 15.11.2018

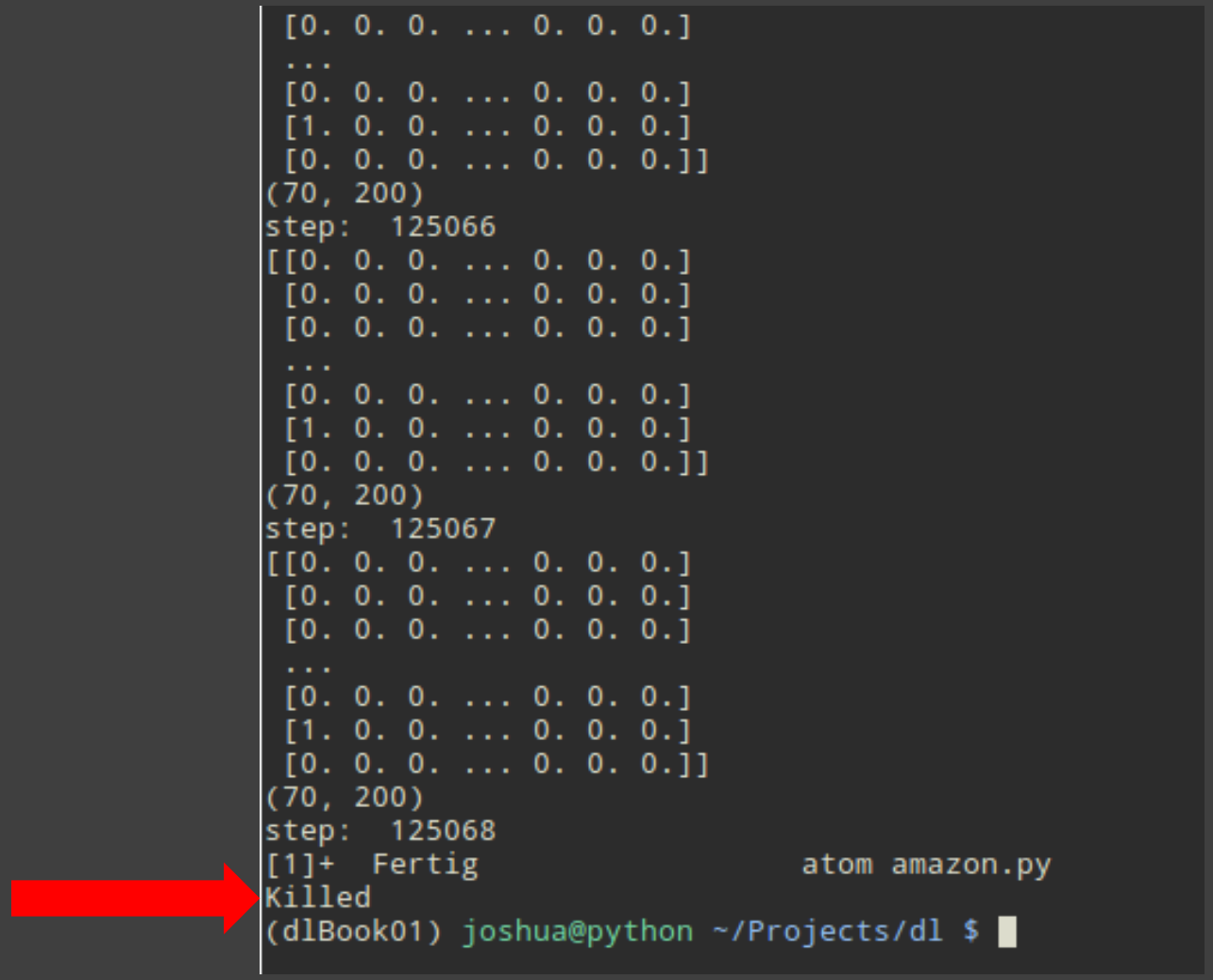
Intro [1]

- Beispielprogramm
- Neuronales Netz
- nicht getestet
- what could go wrong?

```
118  convnet.add(
119      Conv1D(1024, 7, activation="relu")
120  )#8
121  convnet.add(convnet.add(MaxPooling1D(pool_size=(3,3))))#9
122  convnet.add(Flatten())#10
123  convnet.add(Dense(2048, activation="relu")) #11
124  convnet.add(Dense(2048, activation="relu")) #12
125  convnet.add(Dense(1, activation="sigmoid")) #13
126
127  convnet.compile(
128      loss="mean_squared_error", optimizer="sgd", metrics=["accuracy"])
129  convnet.fit(tensor, train_labels[:1001], epochs=5, verbose=1)
130
131  metrics = convnet.evaluate(test_samples, c_test_labels, verbose=1)
132  print("\n%s: %.2f%%" % (convnet.metrics_names[1], metrics[1]*100))
133
```

Intro [2]

- Programmdurchlauf
- Prompt wieder da
- keine Logs?
- Bash killt Programm
- warum?



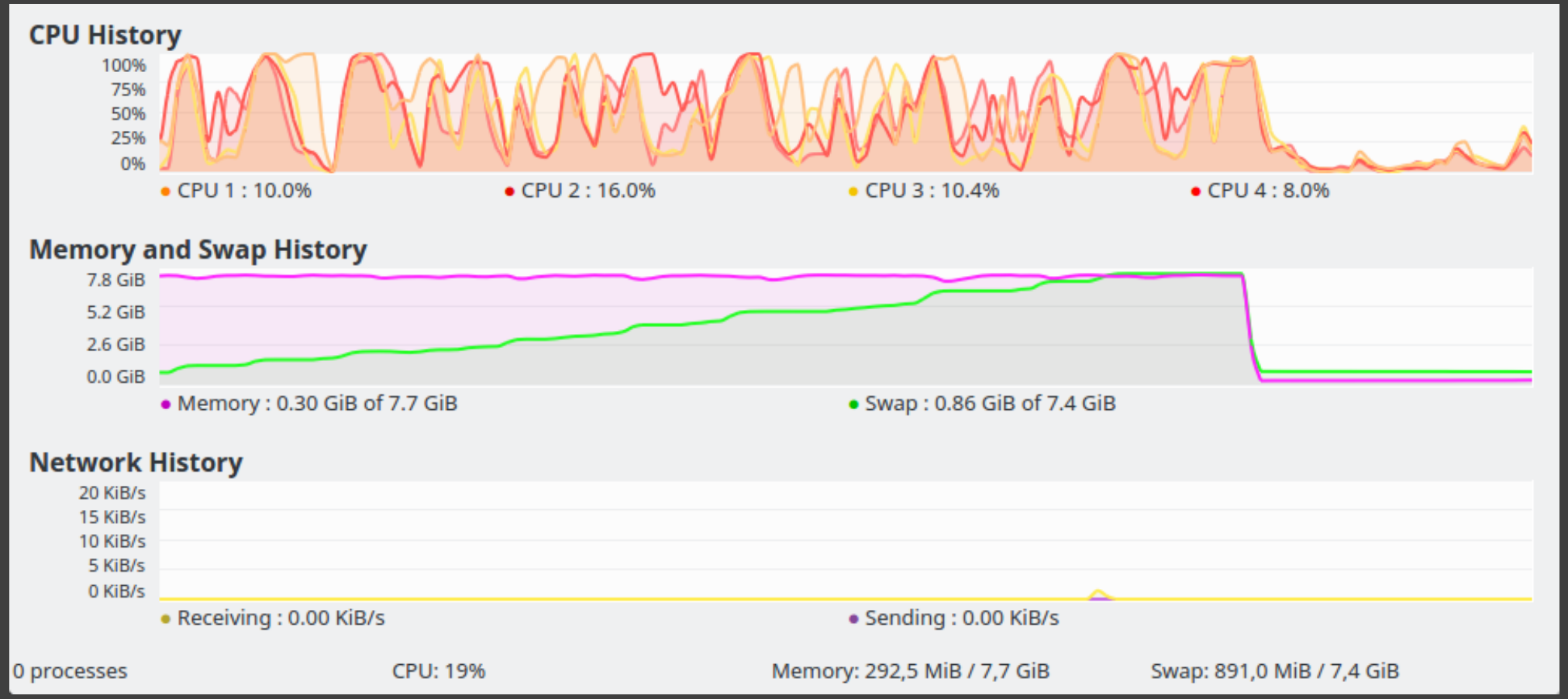
Intro [3]

- Linux /proc-files
- `watch -n 1 cat /proc/meminfo`
- Felder entziffern
- Snapshots
- Unübersichtlich

```
Every 1,0s: cat meminfo

MemTotal:        8042188 kB
MemFree:          132832 kB
MemAvailable:     105004 kB
Buffers:            8180 kB
Cached:           333980 kB
SwapCached:       471676 kB
Active:          6657828 kB
Inactive:        1014288 kB
Active(anon):    6575816 kB
Inactive(anon):   934136 kB
Active(file):      82012 kB
Inactive(file):    80152 kB
Unevictable:          32 kB
Mlocked:              32 kB
SwapTotal:       7811580 kB
SwapFree:        2021512 kB
```

Intro [4]


CPU History
100%
75%
50%
25%
0%
CPU 1 : 10.0%
CPU 2 : 16.0%
CPU 3 : 10.4%
CPU 4 : 8.0%
Memory and Swap History
7.8 GiB
5.2 GiB
2.6 GiB
0.0 GiB
Memory : 0.30 GiB of 7.7 GiB
Swap : 0.86 GiB of 7.4 GiB
Network History
20 KiB/s
15 KiB/s
10 KiB/s
5 KiB/s
0 KiB/s
Receiving : 0.00 KiB/s
Sending : 0.00 KiB/s
0 processes
CPU: 19%
Memory: 292,5 MiB / 7,7 GiB
Swap: 891,0 MiB / 7,4 GiB

Gliederung

- Einleitung
- Visualisierungspipeline
  - Datenerhebung
  - Preprocessing
  - Visualisierung
- Visualisierung im Kontext HPC
- Beispiel: Vampir
- Beispiel: Grafana
- Zusammenfassung
- Quellen

Warum eigentlich visualisieren?

Transformation von Daten in Grafiken hilft bei:

- Explorativer Datenanalyse
- Kognition komplexer Inhalte
- Erklärung von Strukturen und Prozessen [1]

Einleitung [1]

- Kontext: High-Performance-Computing (HPC)
- Ziel: effiziente Ausführung paralleler Programme
- Visualisierung von Leistungsdaten
- Eignet sich für
  - Monitoring
  - Erkennung von fehlerhaftem Verhalten
  - Tool für Performance Analyse

Einleitung [2]

Beispiel: Welche Daten können visualisiert werden?

- CPU-Auslastung
- Speicherverbrauch
- Netzwerkkommunikation
- Hardware Counter
- MPI-Kommunikation

Einleitung [3]

„Eine Visualisierung ist dann gut, wenn die bildliche Darstellung das kommunikative Ziel erreicht hat[…]“

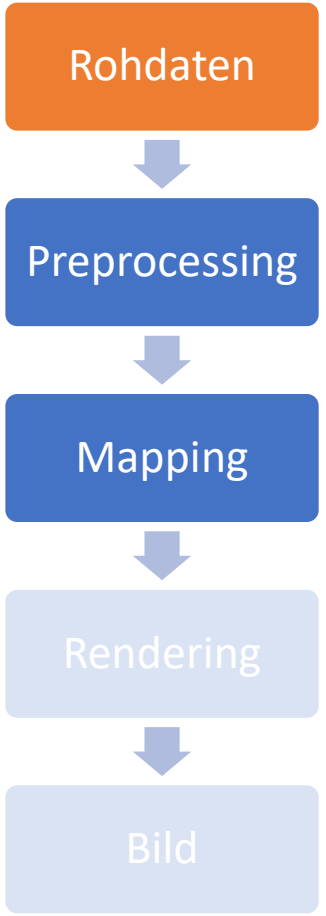
Prof. Dr. Fischer-Stabel in [1]

Visualisierungspipeline

Abb. 1: Visualisierungspipeline nach [1]

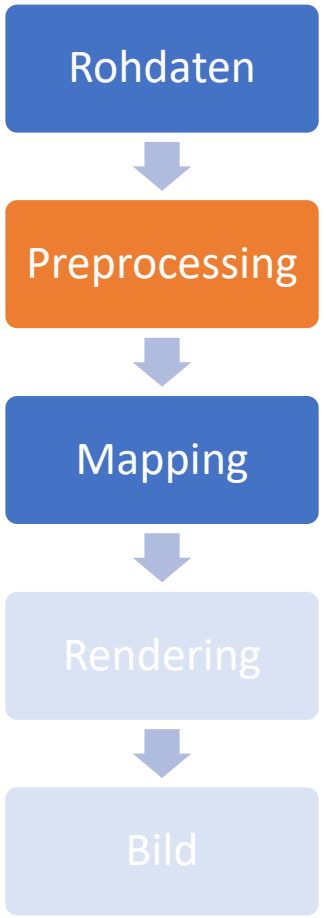
Datenerhebung

- Daten manuell speichern
- `/proc`-Files auslesen
- Profiling/Tracing-Tools verwenden
- Datenerhebung zeit- oder eventbasiert
- ggf. manuelle Instrumentierung erforderlich
- Beispiel: Vampir Trace, Score-P



Preprocessing

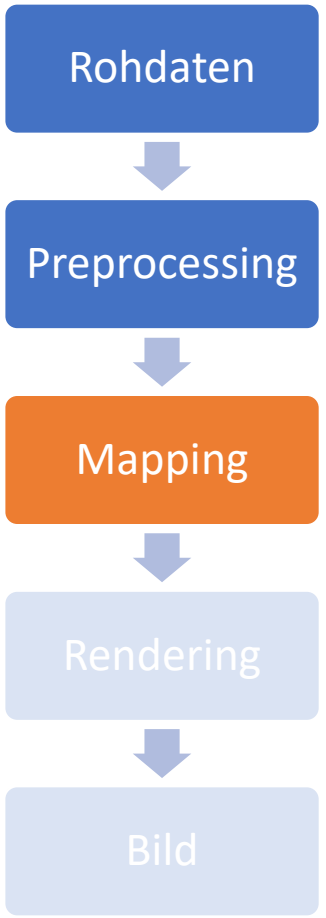
- Behandlung fehlender Datensätze
- Konvertierung von Formaten
- Bereinigung der Daten
- Aggregation und Reduktion der Daten
- Klassifizieren und Clustering



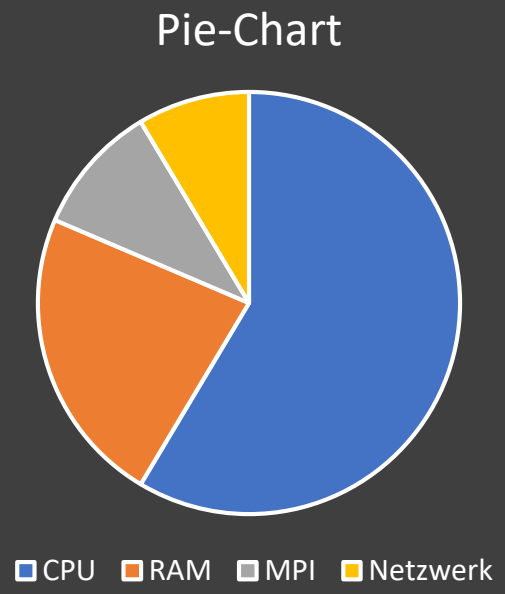
Visualisierung

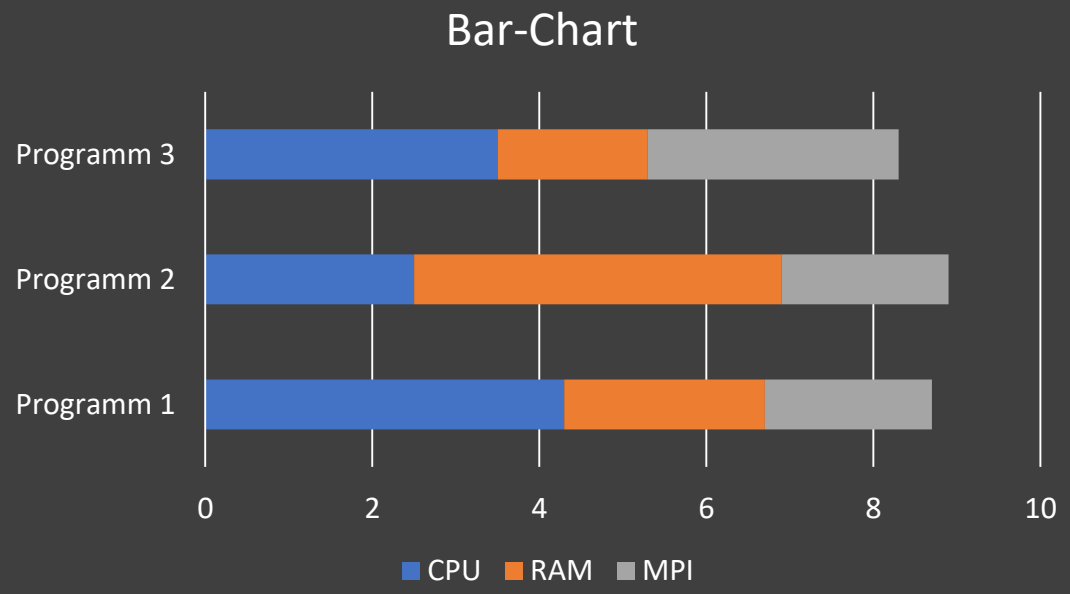
Zu beachten

- Struktur der Daten
- Ziel der Visualisierung
- Fähigkeiten und Eigenschaften der Zielperson
- Kontext der Darstellung



Verteilungen [1]



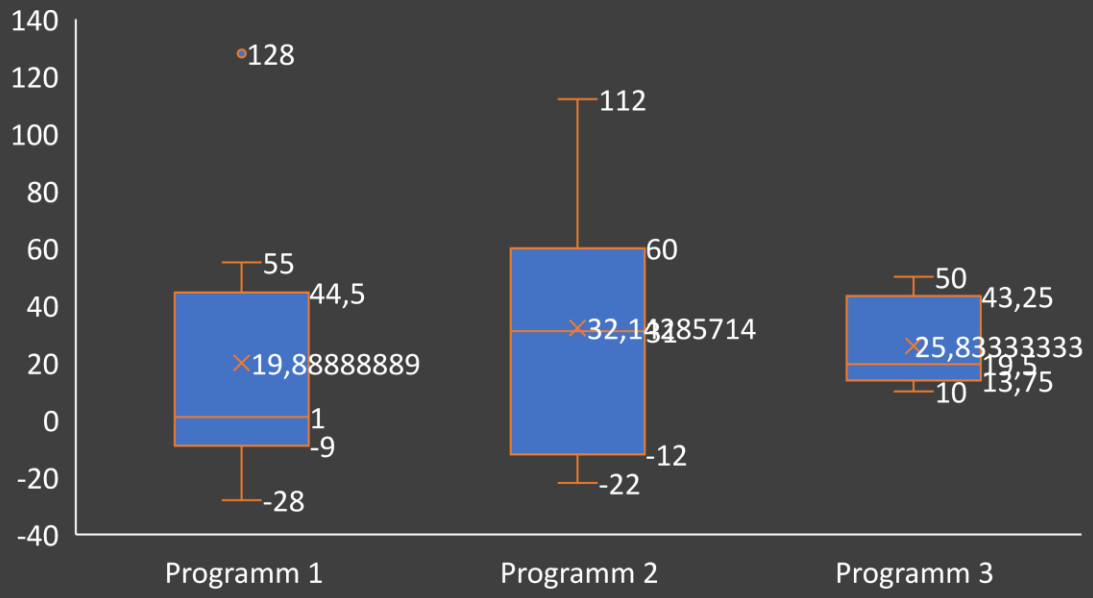


Use-Case: Verbrauch von Systemressourcen

Verteilungen [2]

Box-/Whiskerplot [8]

- Quartile 1 - 3
- min/max Werte | 1.5x IQR
- Ausreißer
- Mindestens ordinalskaliert
- Use-Case: Erkennung von Varianz in Ausführungsdauer



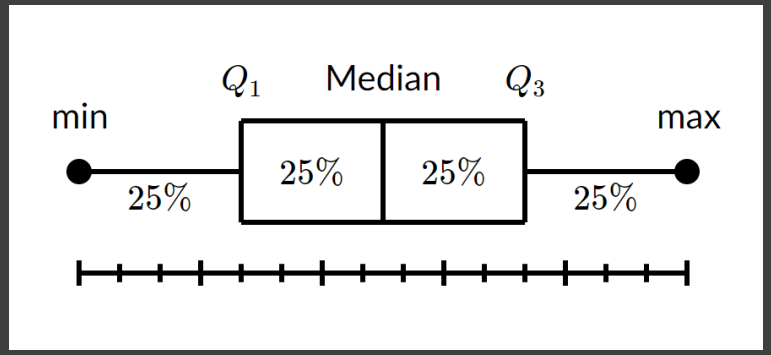
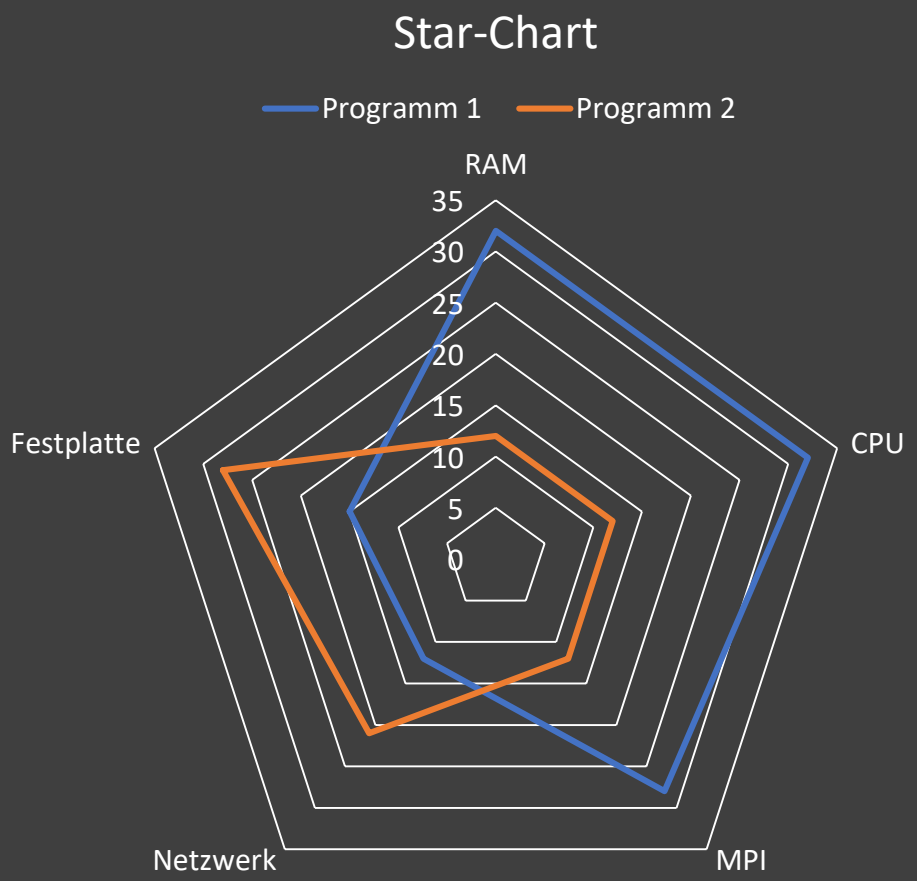


Abb. 16: Whisker-Plot

Parameterausprägung

Star-Chart

- Ausprägung verschiedener Parameter
- unterschiedliche Skalen möglich
- Use-Case: Vergleich von Profilen



Timing

Gantt-Chart

- Zeigt Dauer, zeitliche Abstände und Dependencies
- Nützlich für Timing
- Use-Case: Aufrufzeitpunkt von Funktionen

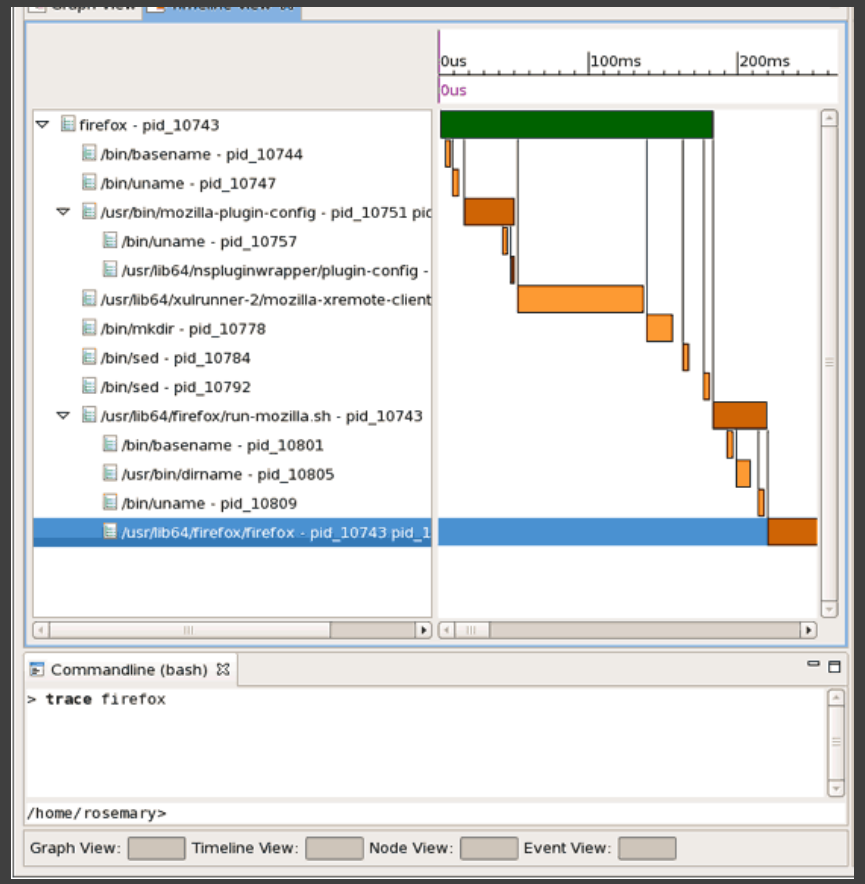
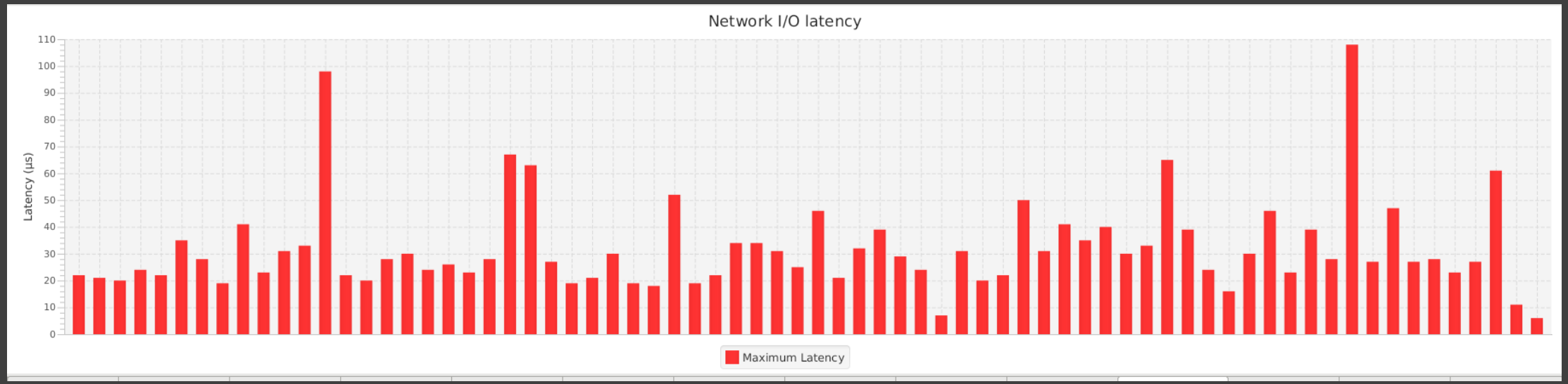


Abb. 17: Gantt-Chart

Zeitverlauf



Use-Case: visualisiert Veränderung über Zeit

Abb. 18: Timeline

Kontext: HPC-Visualisierung [1]

- Ziel: Parallele Programme optimieren
- Beispiel Mistral: 3,300 Nodes (24 & 36 Kerne)
- Beispielrechnung: 100 ints (4bytes) pro Sekunde loggen
  - Anwendung läuft auf 700 Nodes
  - Auf jedem Node laufen 24 Prozesse
  - Logging für eine Stunde

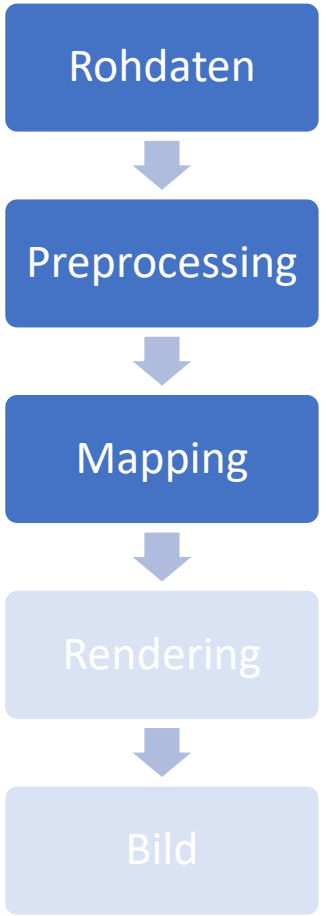
→ `4*100*24*700*60*60 ≈ 22.53 GiB/h`

Problem: große Datenmenge und asymmetrischen Systemstrukturen

Kontext: HPC-Visualisierung [2]

Mit Blick auf die Visualisierungspipeline

- Speichern der Daten in Datenbanken
- Externes Programm übernimmt Visualisierung
- Aggregation und Reduktion essentiell
- Daten in Hierarchien und Abstraktionsebenen aufteilen
- Interaktionstechniken ermöglichen Navigation (z.B. zoomen)



Beispiel: Vampir

Visualisierung für die Performance-Analyse

Vampir Mastertimeline

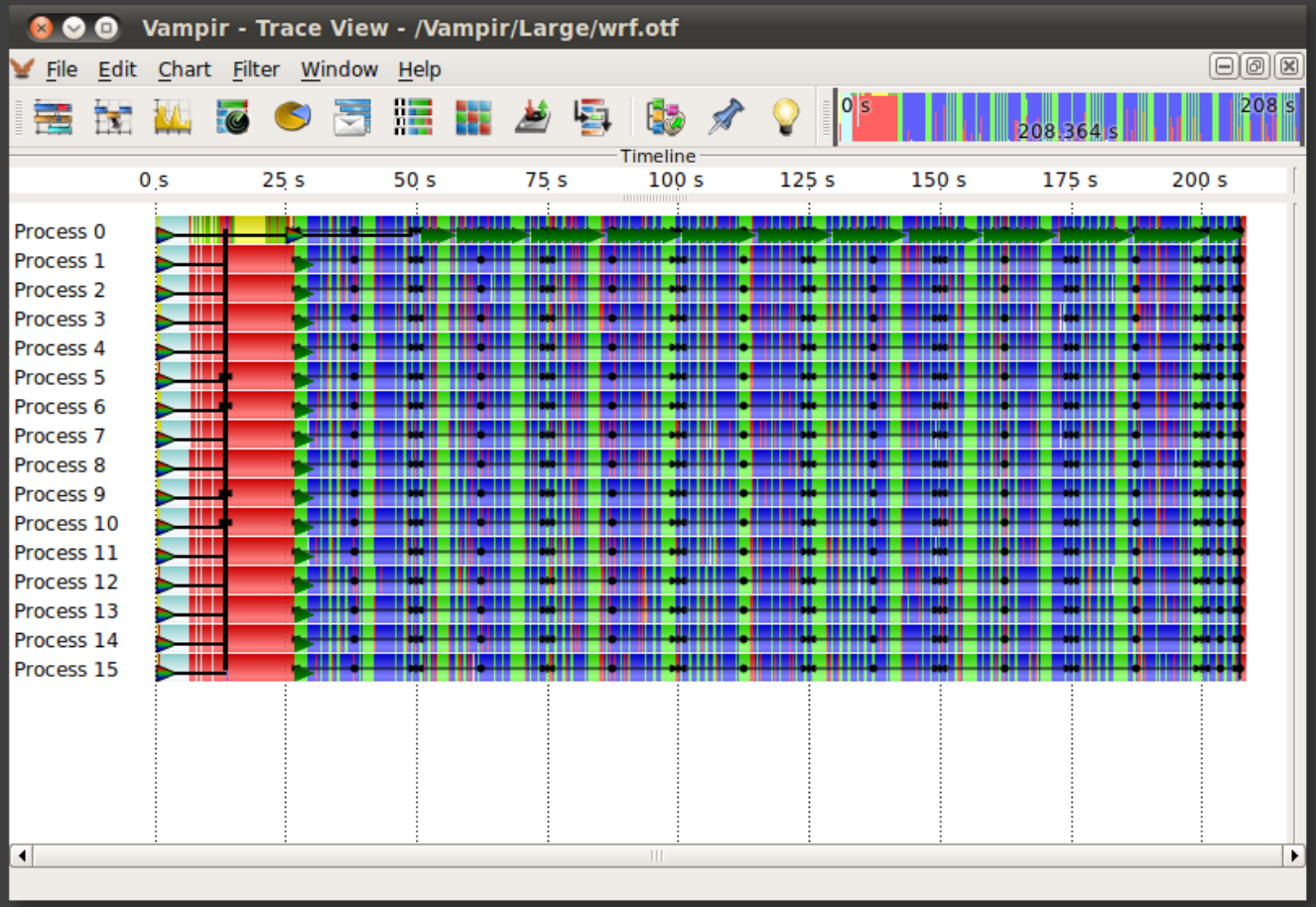


Abb. 3: Vampir Mastertimeline aus [6]

Skalierbarkeit in Vampir

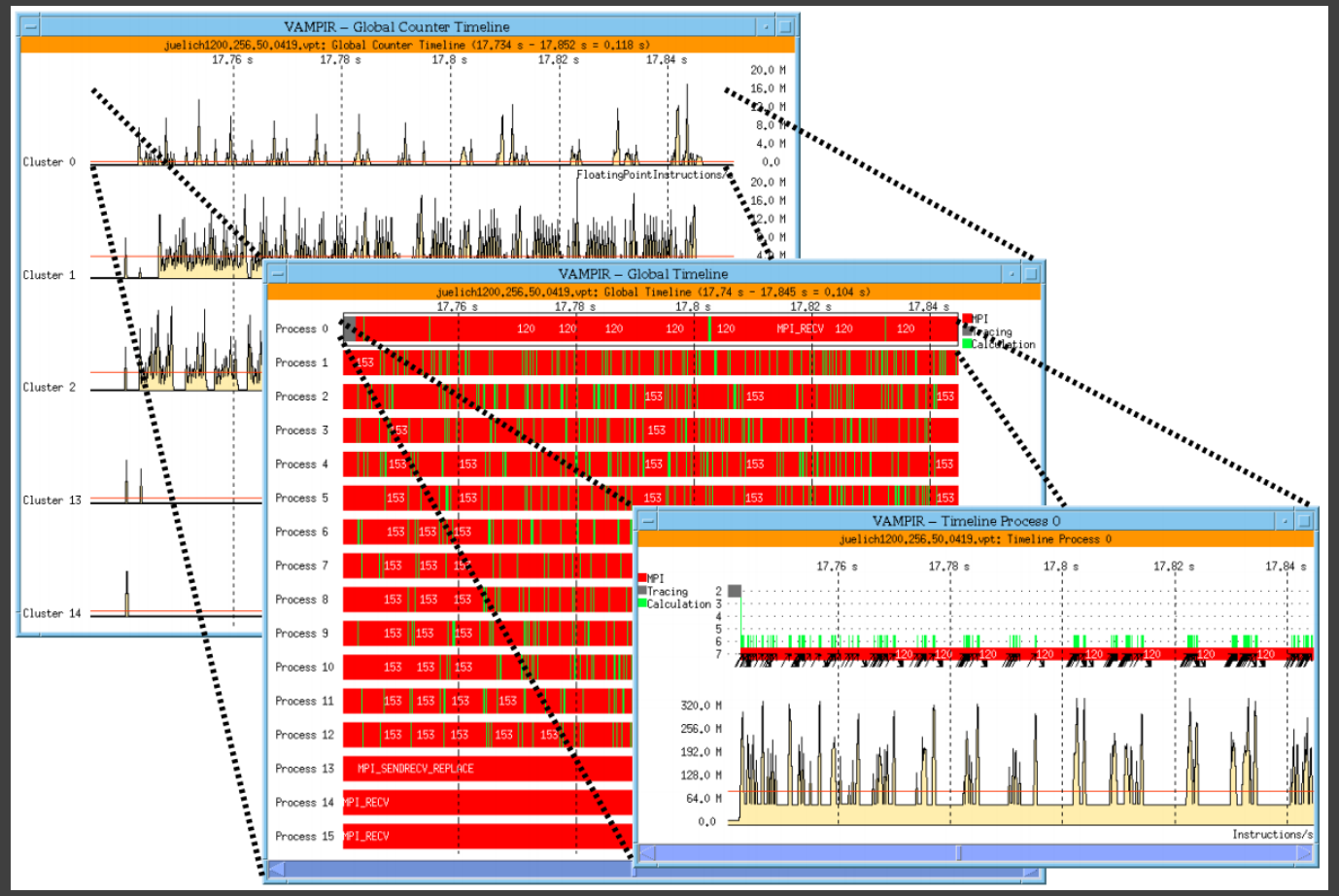


Abb. 4: Vampir Hierarchie aus [2]

Beispiel: Abstraktionsebenen [1]

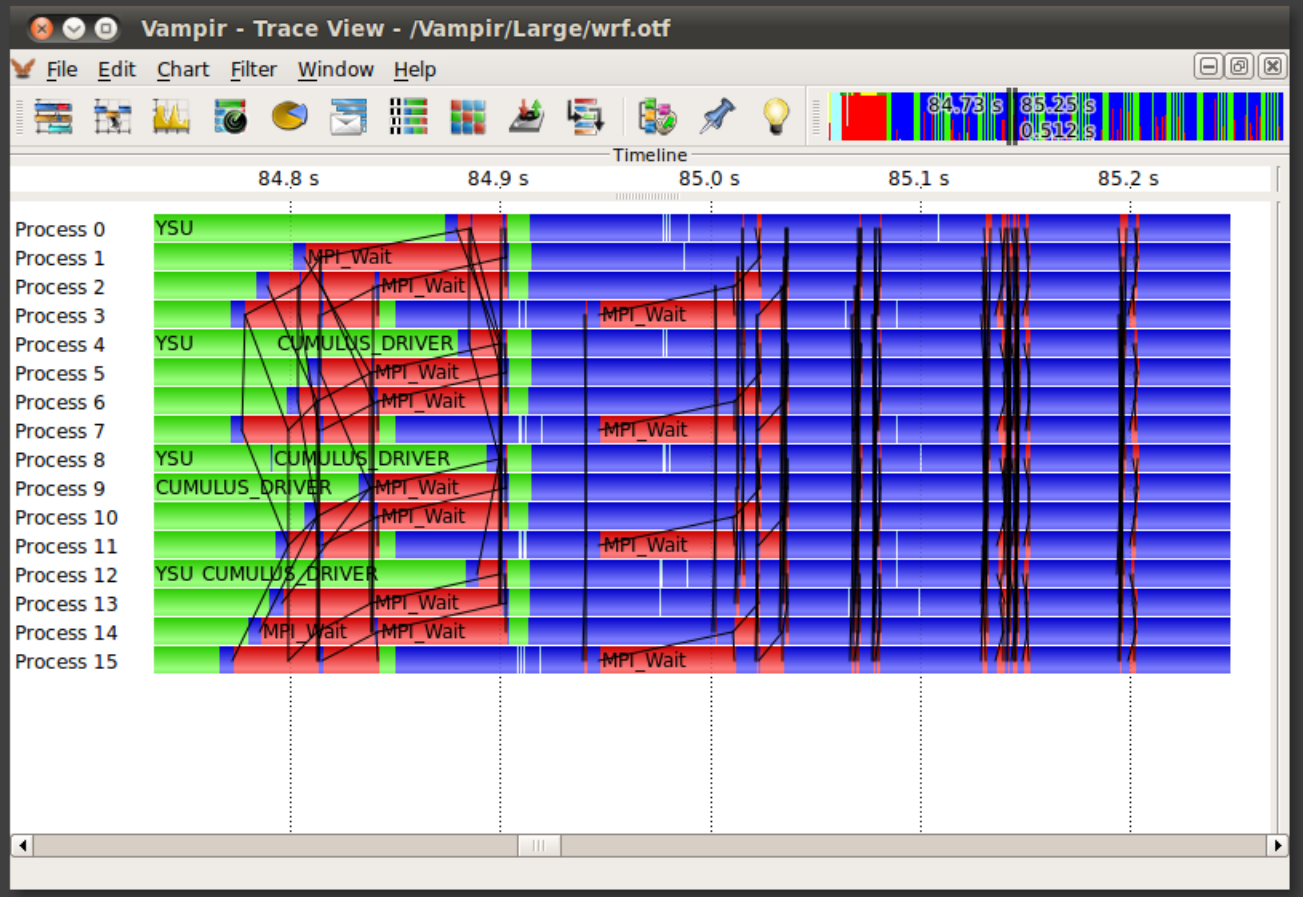


Abb. 6: Vampir Zoom aus [6]

Beispiel: Abstraktionsebenen [2]

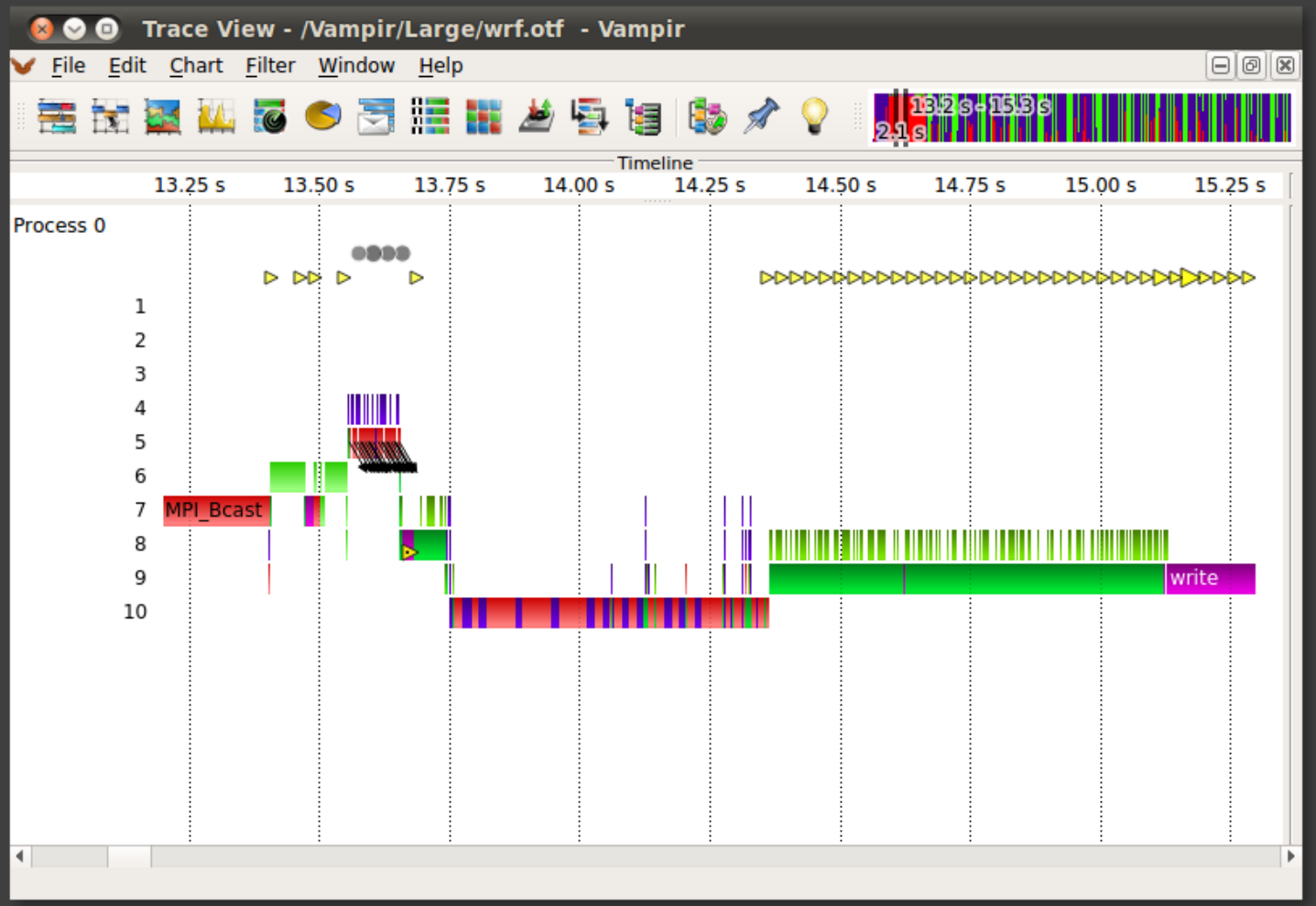


Abb. 7: Vampir Zoom aus [6]

Beispiel: Optimierung mit Vampir

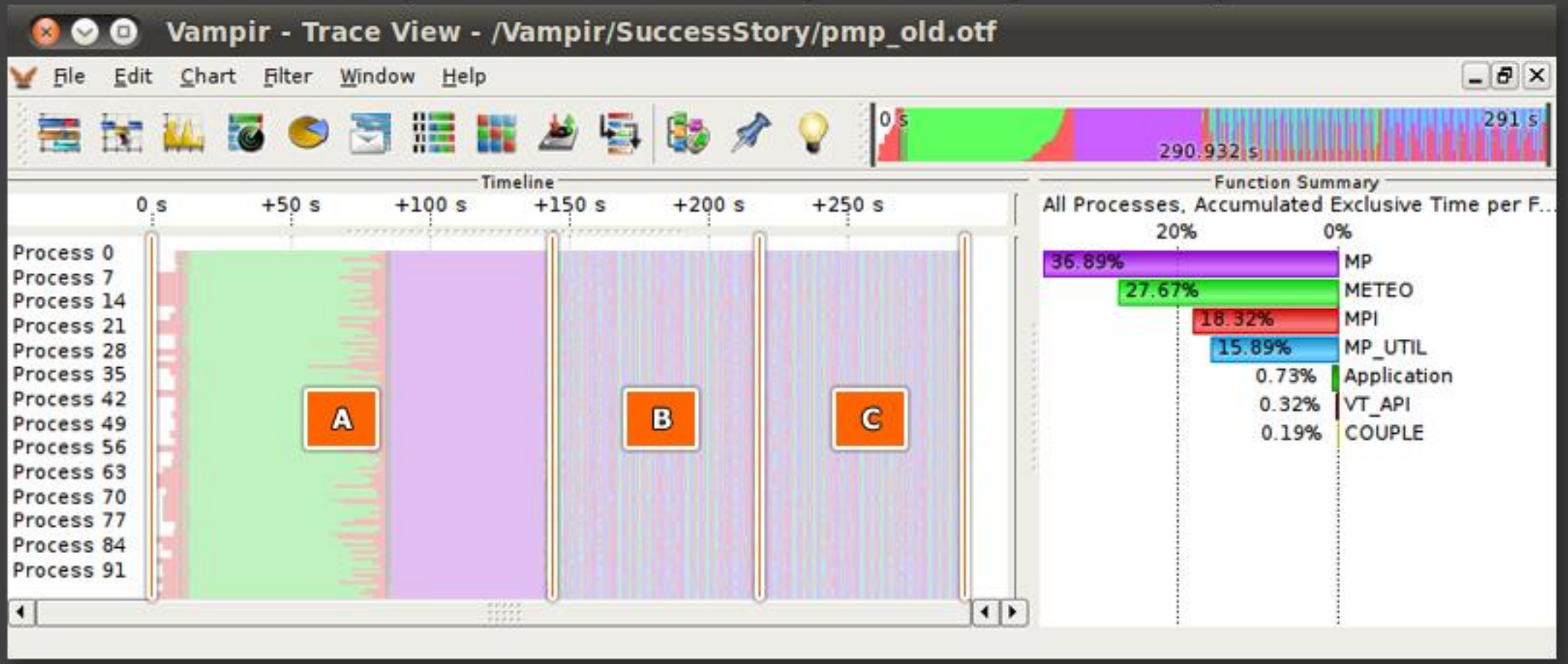


Abb. 8: Vampir Optimierung aus [6]

Beispiel: Optimierung mit Vampir

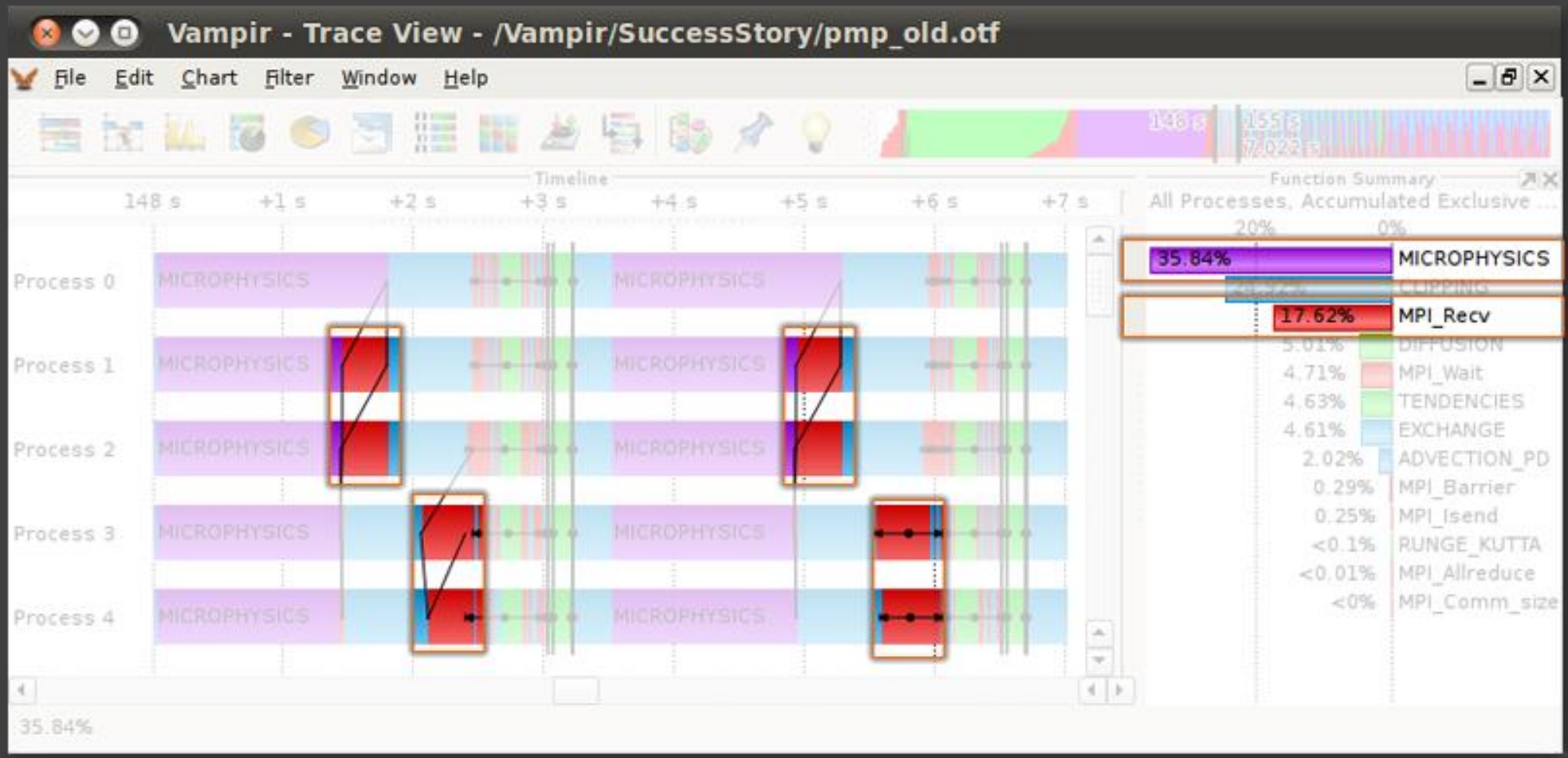


Abb. 9: Vampir Optimierung aus [6]

Beispiel: Optimierung mit Vampir

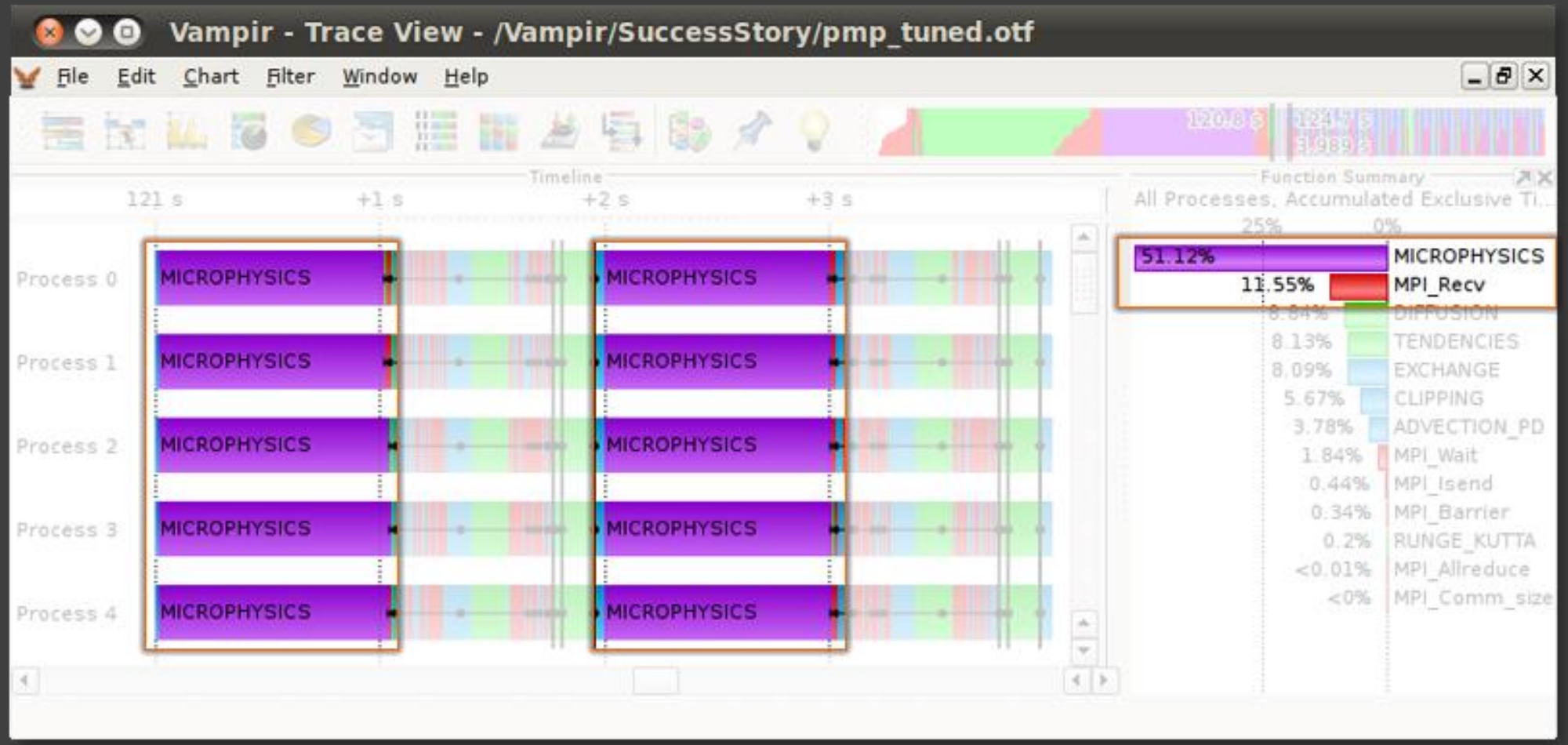


Abb. 10: Vampir Optimierung aus [6]

Beispiel: Optimierung mit Vampir

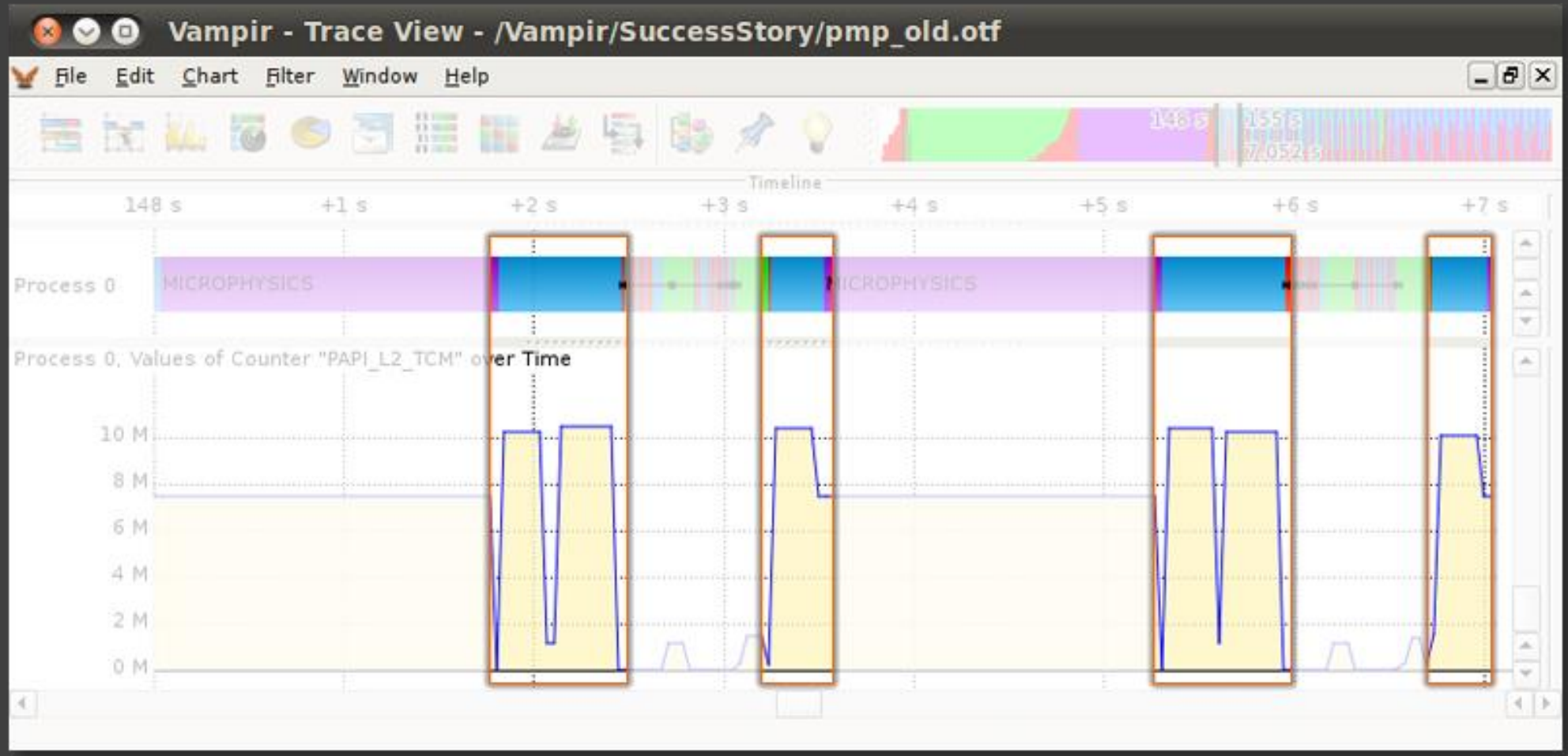


Abb. 11: Vampir Optimierung aus [6]

Beispiel: Optimierung mit Vampir

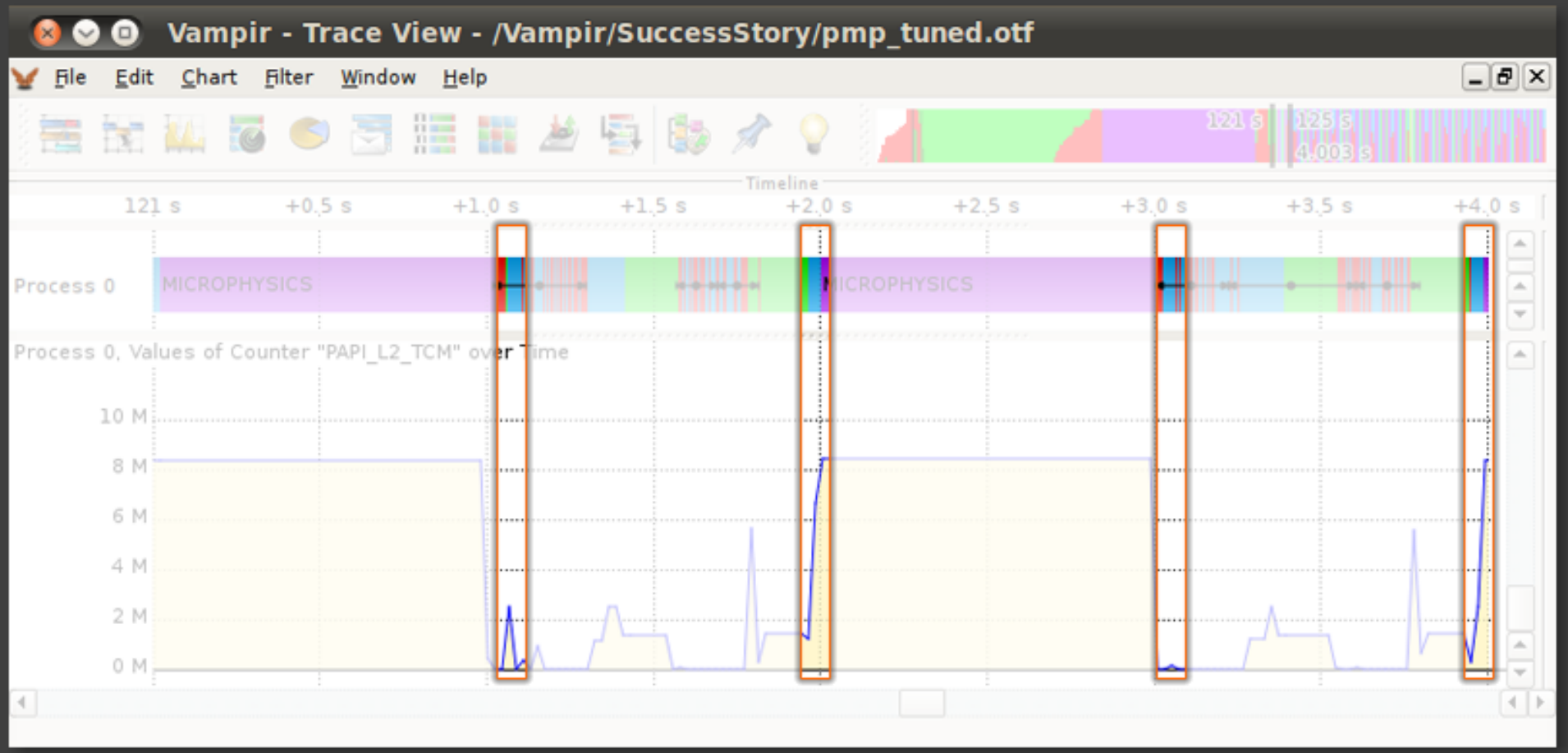


Abb. 12: Vampir Optimierung aus [6]

Beispiel: Optimierung mit Vampir

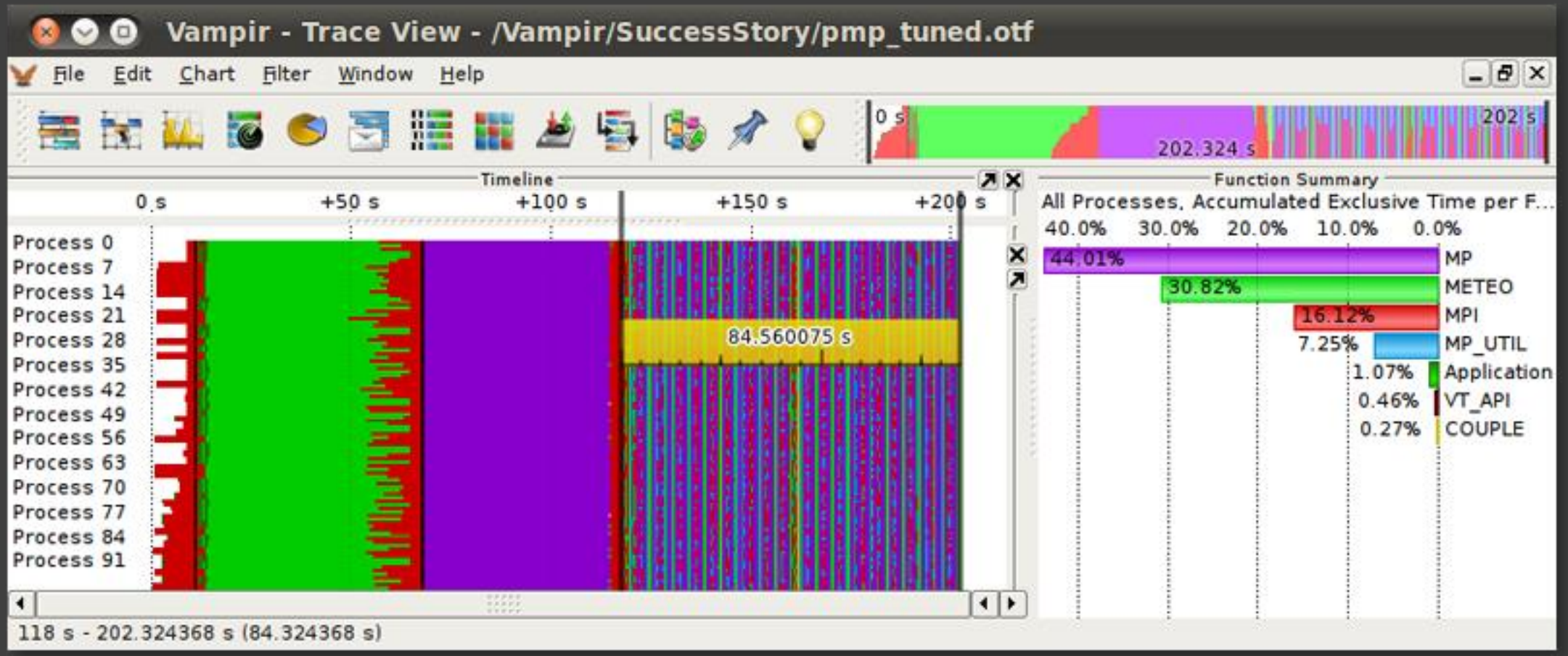


Abb. 13: Vampir Optimierung aus [6]

Beispiel: Grafana

Visualisierung für das Monitoring

Loginnodes

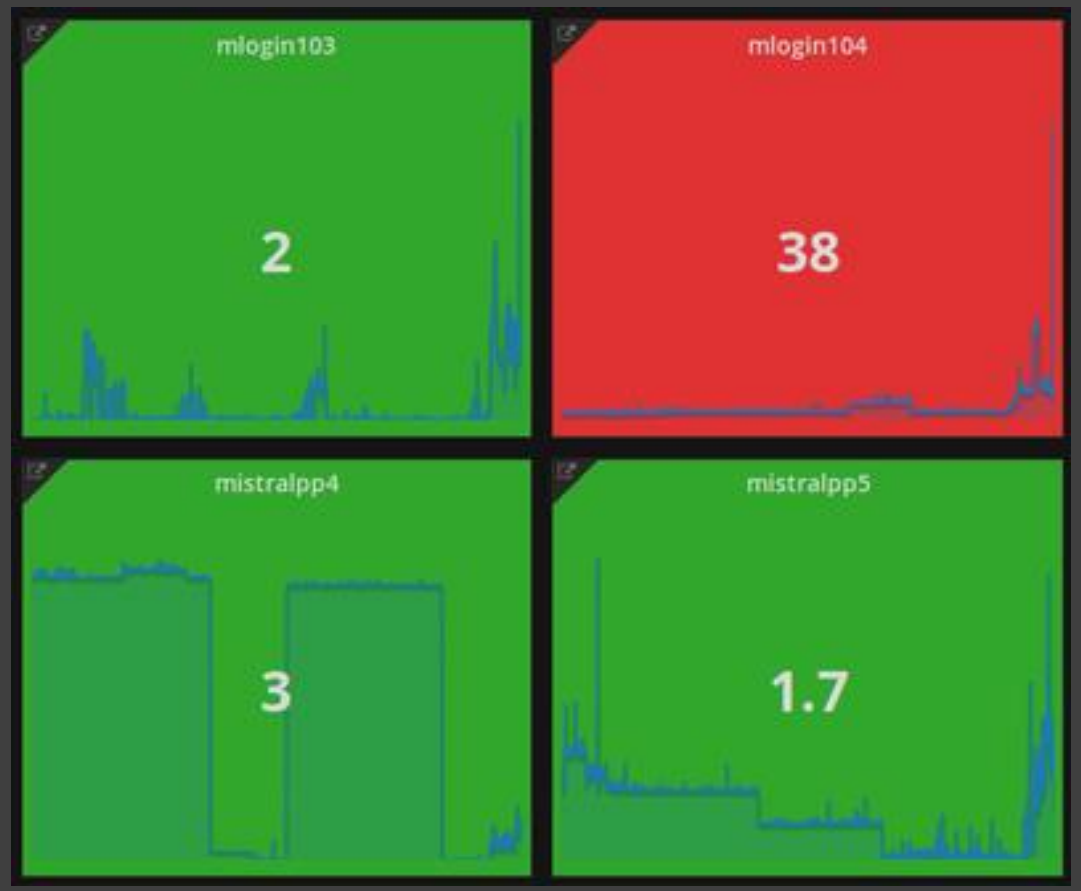


Abb. 14: Grafana Login aus [7]

Beispiel: Panel

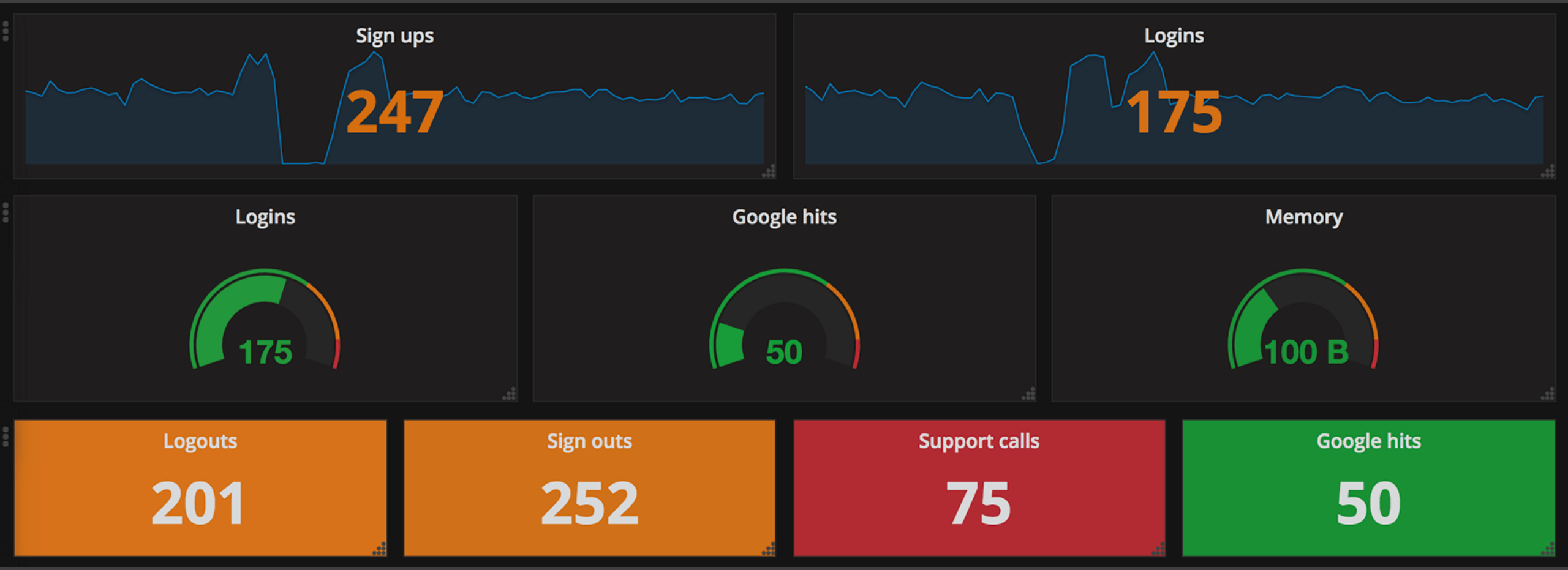


Abb. 15: Grafana Panel

Beispiel: Dasboard

Abb. 16: Grafana Dashboard

Beispiel: einzelner Graph

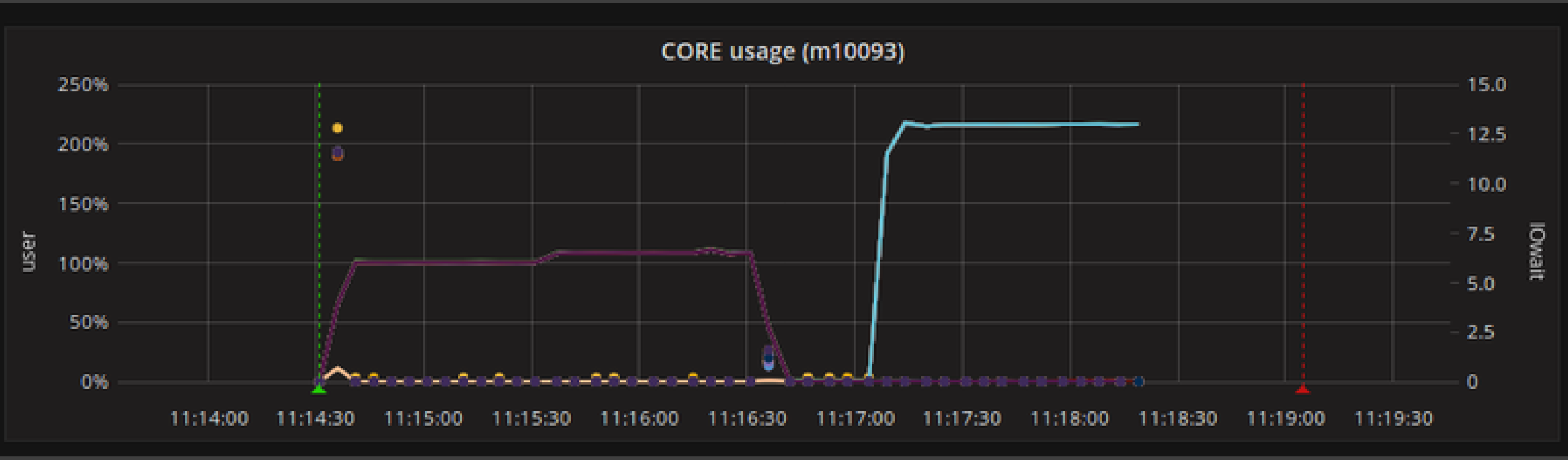


Abb. 17: Grafana Graph aus [7]

Zusammenfassung

- Visualisierung erleichtert Wahrnehmung, Kommunikation und Verständnis komplexer Daten
- Ermöglicht explorative Perfomance Analyse bei parallelen Programmen
- Pipeline der Visualisierung
- Skalierbarkeit durch hierarchische Strukturen und Abstraktion möglich

Bildquellen

Abb. 1: Wikipedia (2018). Retrieved from https://de.wikipedia.org/wiki/Gantt-Diagramm (As of: 14.11.2018)

Abb. 2, 4-12: GWT-TUD GmbH (2018). Retrieved from https://vampir.eu/ (As of: 14.11.2018)

Abb. 3: Brunst, H., Winkler, M., Nagel, W. E., & Hoppe, H. C. (2001, May). Performance optimization for large scale computing: The scalable VAMPIR approach. In International conference on computational science (pp. 751-760). Springer, Berlin, Heidelberg.

Abb. 13, 16: DKRZ (n. Y.). Retrieved from https://www.dkrz.de/up/de-systems/de-monitoring (As of: 14.11.2018)

Abb. 14,15: Grafana Labs (2018). Retrieved from https://grafana.com/ (As of: 14.11.2018)

Abb. 16: Khan Academy (2018). Retrieved from https://de.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review (As of: 14.11.2018)

Abb. 17-18: Allexus Breeze (2018). Retrieved from https://www.ellexus.com/products/breeze-hpc-file-system-profiling/ (As of: 14.11.2018)

Quellen

[1] Fischer-Stabel, P., & . (2018). Datenvisualisierung: Vom Diagramm zur Virtual Reality. München: UVK Verlag (pp. 21-60, 191-204).

[2] Brunst, H., Winkler, M., Nagel, W. E., & Hoppe, H. C. (2001, May). Performance optimization for large scale computing: The scalable VAMPIR approach. In International conference on computational science (pp. 751-760). Springer, Berlin, Heidelberg.

[3] Sunderland, A., & Porter, A. R. (2007). Profiling parallel performance using vampir and paraver.

[4] Drebes, A., Pop, A., Heydemann, K., & Cohen, A. (2016, April). Interactive visualization of cross-layer performance anomalies in dynamic task-parallel applications and systems. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 274-283).

[5] Weaver, C., Barr, K. C., Marsman, E., Ernst, D., & Austin, T. (2001). Performance analysis using pipeline visualization. In 2001 IEEE International Symposium on Performance Analysis of Systems and Software (pp. 18-21). IEEE.

[6] GWT-TUD GmbH (2018). Retrieved from https://vampir.eu/ (As of: 14.11.2018)

[7] DKRZ (n. Y.). Retrieved from https://www.dkrz.de/up/de-systems/de-mistral (As of: 14.11.2018)

[8] Wikipedia (2018). Retrieved from https://de.wikipedia.org/wiki/Box-Plot (As of: 14.11.2018)