Profiling von Software-Energieverbrauch

Seminar „Ausgewählte Kapitel der Systemsoftwaretechnik: Energiegewahre Systemsoftware“ im Sommersemester 2013

Michael Fiedler

6. Juni 2013

Motivation (1)

- Grundproblem bei Rechensystemen: zu hoher Energieverbrauch
  - Batteriekapazität wichtiger beschränkender Faktor für mobile Geräte (Laufzeit, Kosten)
  - Abwärme und Stromkosten auch bei größerer Hardware relevant
- Wie kann der Energieverbrauch gesenkt werden?
  - Möglichkeiten
    - sparsamere Hardware
    - **Software muss sparsamer mit Energie umgehen**
  - Entwickler muss erkennen können:
    - Wo in der Software sind kritische Stellen bzgl. des Energieverbrauchs?
    - Welche Auswirkungen haben meine Änderungen an der Software auf deren Energieverbrauch?
  - damit: Umsetzung gezielter Energieverbrauchsoptimierung möglich

Motivation (2)

Hilfe durch **Profiling von Software-Energieverbrauch**:

Leitfrage

Wo genau in meiner Software wird **wieviel** Energie verbraucht?

Überblick

Entwurfsaspekte beim Profiling

Beispiele für Profiler-Umsetzungen

Zusammenfassung

Überblick

Entwurfsaspekte beim Profiling

Beispiele für Profiler-Umsetzungen

Zusammenfassung

Grundlegende Vorgehensweise

Bestandteile beim Profiling von Software-Energieverbrauch:

1. **Programm** durchlaufen
2. bei Ausführung benötigten **Energieverbrauch** ermitteln
3. Energieverbrauch und Programmstruktur **verknüpfen**

Entwurfsaspekte beim Profiling: Überblick

- Granularität
- Messung des Energieverbrauchs
- Zuordnung Energieverbrauch $\leftrightarrow$ Programmstruktur
- Programmpfadabdeckung

Granularität

- Für „einfache“ Anwendungssoftware
  - Prozess
  - Thread
  - Funktion
  - Systemaufruf
- Auf höherer Ebene u. a.
  - Rechner (*system-level profiling*)
  - Sensornetzwerke
  - Cloud-Computing

Messung des Energieverbrauchs

- Instrumentierung der Hardware
- Verwendung von Hardware-Sensoren (hinreichend verfügbar?)
- Modellierung des Energieverbrauchs
- Ersatzmetriken
  - Batterieladezustand
  - Ereigniszähler (*event counters*)

Zuordnung des Energieverbrauchs

- **Synchroner Energieverbrauch**
  - einfacher Fall (z. B. CPU, RAM)
  - nahe am klassischen Profiling wie mit *gprof*
- **Asynchroner Energieverbrauch**
  - komplizierter
  - nach abgeschlossener Geräteaktivität: vorübergehendes Verweilen in Nicht-Schlafzustand (*tail power state*)
    $\rightarrow$ Schweifenergie (*tail energy*)
  - Aktivitätssperren (*wakelocks*)
  - Geräte mit besonderem Aktivitätsverhalten wie Kamera, GPS, Beschleunigungsmessgerät usw.
  - Berücksichtigung von Gerätezuständen

Asynchr. Energieverbrauch: Schweifenergie (1)

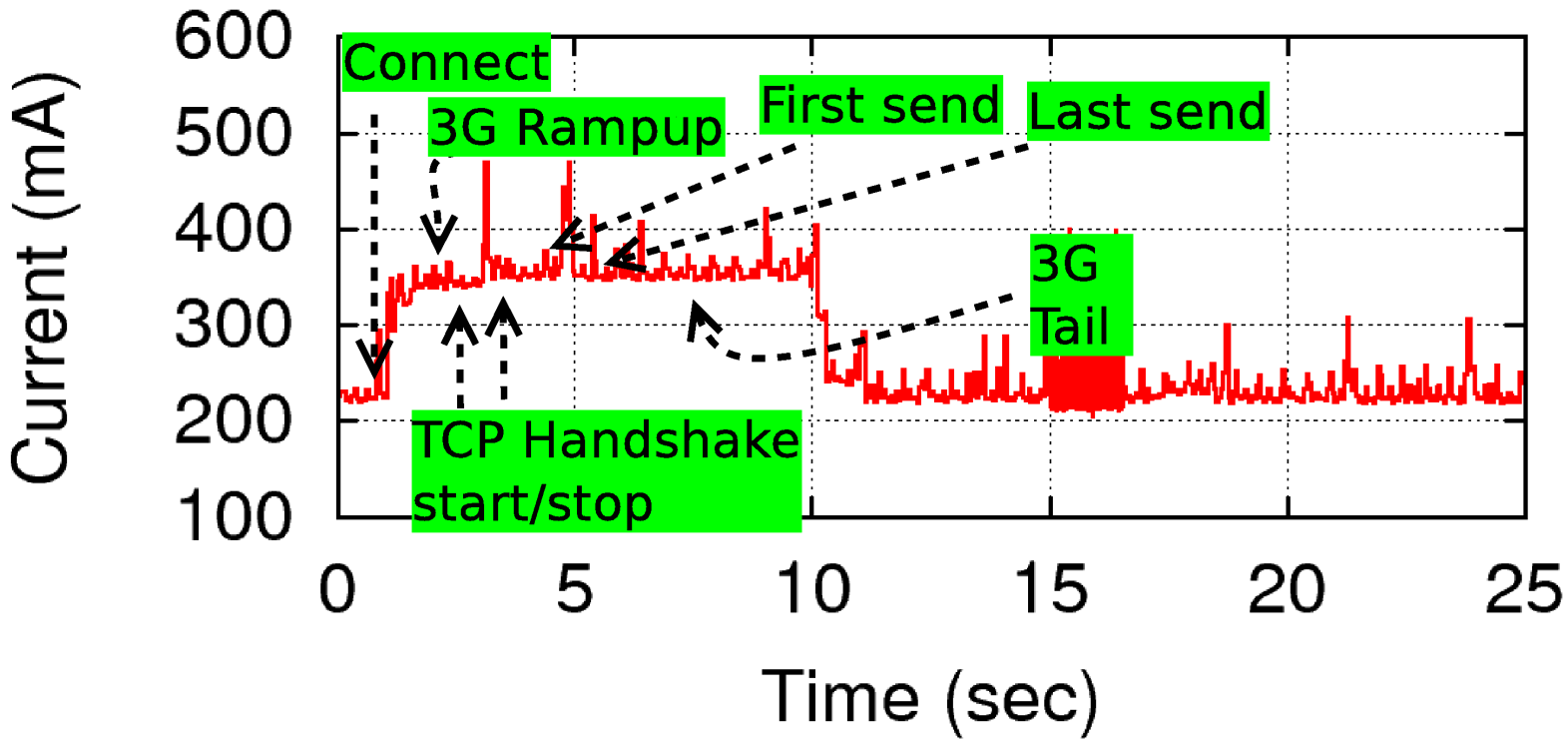


Abbildung: Stromverbrauch beim Senden von 5 Dateneinheiten zu je 10 kB über Mobilfunk (3G) mit *Sendebefehl direkt nach Verbindungsaufbau*. Quelle: [1]

Asynchr. Energieverbrauch: Schweifenergie (2)

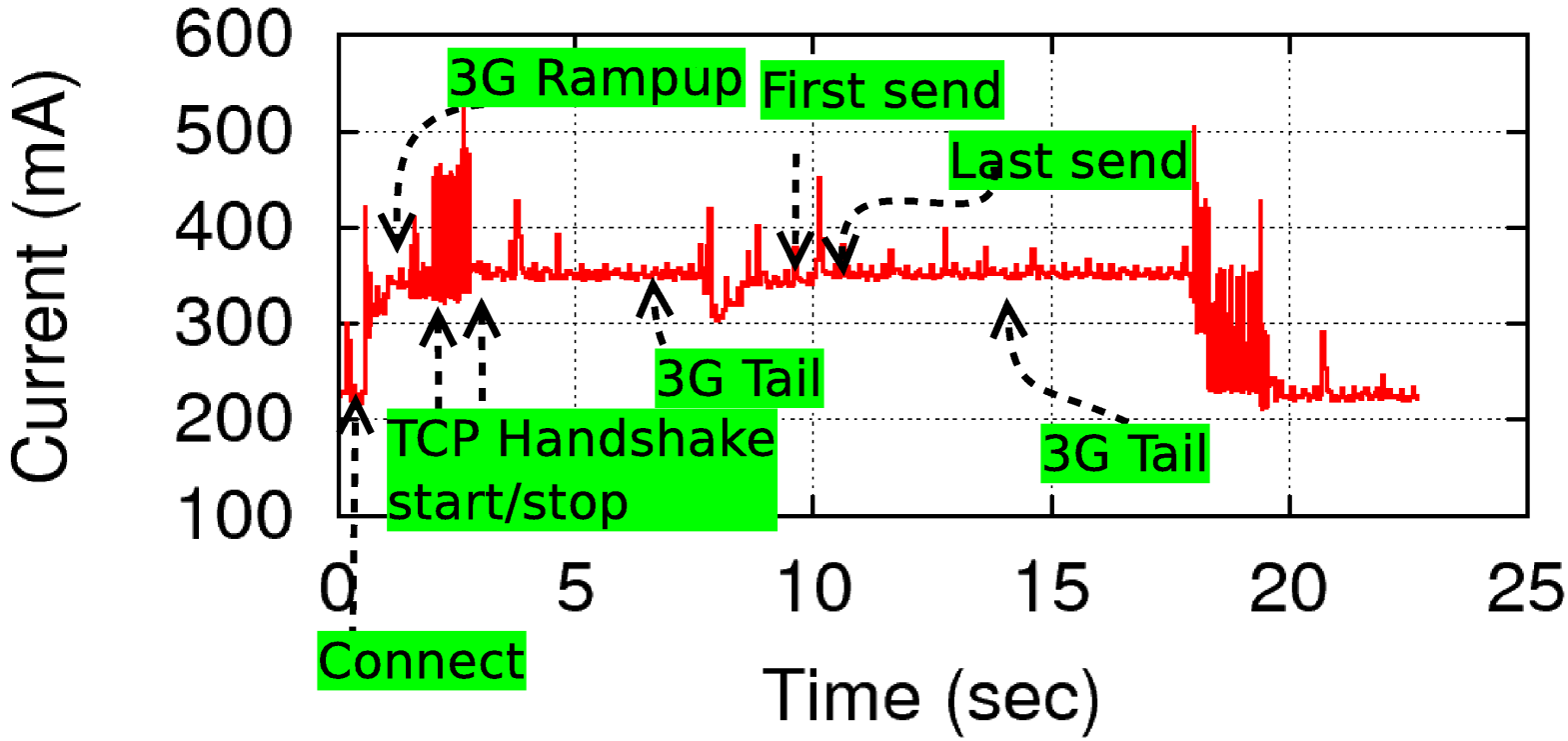


Abbildung: Stromverbrauch beim Senden von 5 Dateneinheiten zu je 10 kB über Mobilfunk (3G) mit *Sendebefehl 5 Sekunden nach Verbindungsaufbau*. Quelle: [1]

Asynchroner Energieverbrauch: Zuordnung (1)

- Verschiedene Zuordnungsstrategien möglich:
  - Aufteilung auf **alle beteiligte Programmeinheiten** (auch mit Gewichtung möglich)
  - Zuweisung zu **letzter Programmeinheit**, die die Schweifenergie nach sich gezogen *hätte*
  - Zuweisung zu **erster Programmeinheit**, die den Schweifenergie nach sich gezogen *hätte*
- Verbergen aber Information...

Asynchroner Energieverbrauch: Zuordnung (2)

- Problem insbesondere der letzten beiden Strategien:
  - eine einzige verantwortliche Programmeinheit erhält Energieverbrauch zugewiesen
  - intuitiv bei Betrachtung des Energieprofils im Nachhinein: *„Entfernung dieser Programmeinheit führt zu vollständiger Beseitigung des Energieverbrauchs“*
  - aber: Energieverbrauch bleibt!

Fazit

Nicht lösbar bzw. asynchrones Verhalten schlecht zu abstrahieren!

- Möglicher Umgang damit:
  - dem Programmierer die Zuteilungsstrategie bewusst machen
  - Schweifenergie getrennt von restlicher Energie verbuchen

Programmpfadabdeckung

- Problem:
  - Ausführungsläufe beim Profiling repräsentativ?
  - bei anderem Testlauf ggf. völlig anderes Energieverbrauchsverhalten
- Lösung mit **symbolischer Ausführung** (vgl. SEEP[2])
  - Ermittlung von möglichen **Programmpfade**
  - Ermittlung der **Bedingungen** für das Durchlaufen der Pfade
  - für Profiling Erstellung mehrerer Binärdateien pro Programmpfad

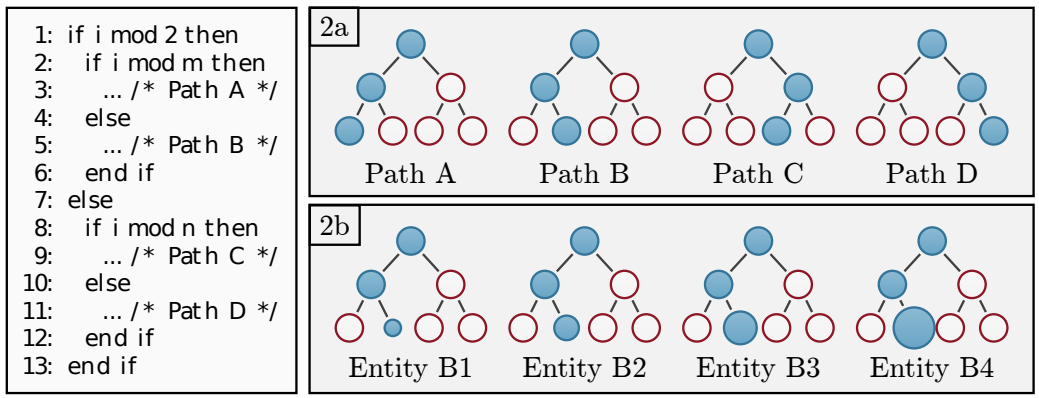


Abbildung: Programmpfade und Programmeinheiten bei SEEP.
Quelle: [2]

Überblick

PowerScope

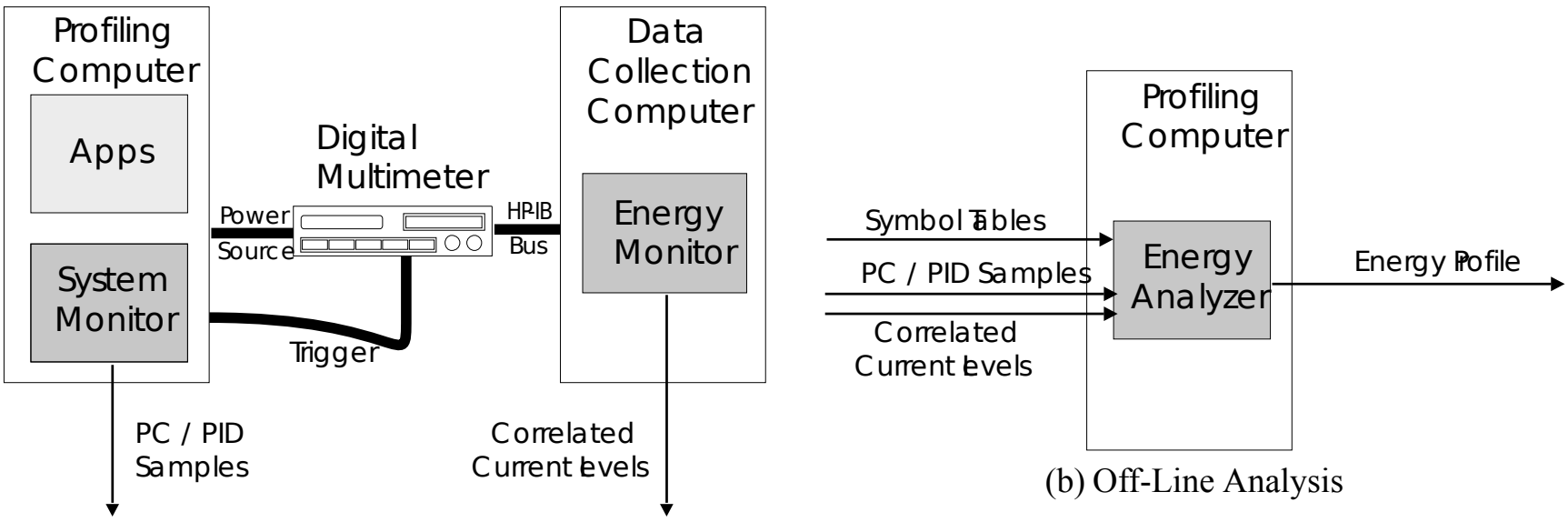
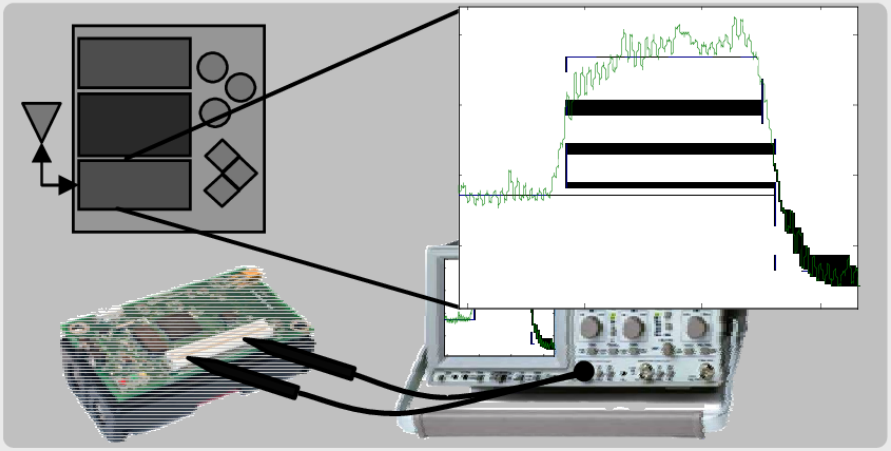


Abbildung: Überblick über PowerScope (1999). Quelle: [3]

AEON (1)

Energy Model:
- Mica2 sensor node
- One model for each hardware component
- Calibration
- Validation

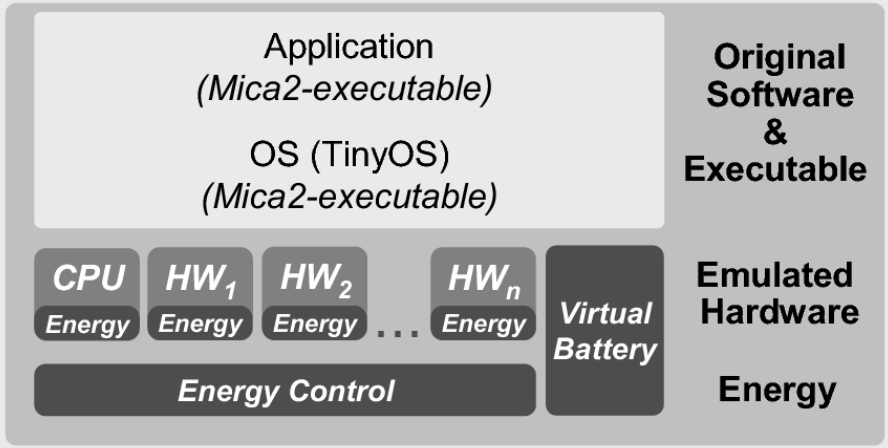


Modeling energy consumption of sensor node components

Abbildung: Überblick über AEON (2005). Quelle: [4]

AEON (2)

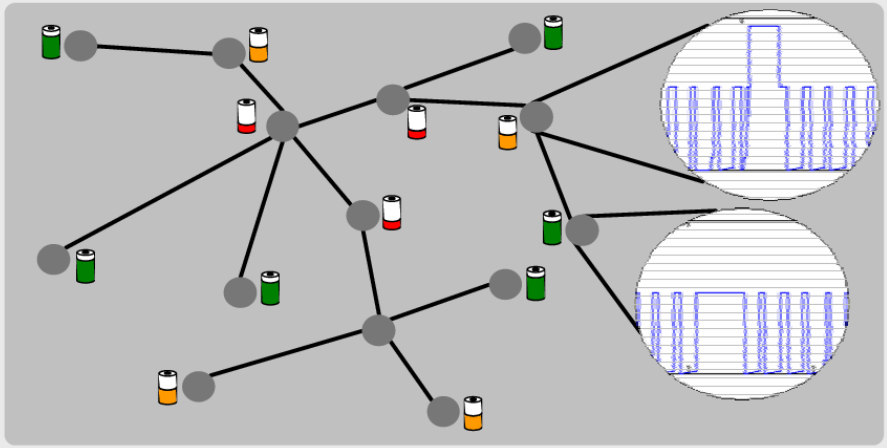
Implementation:
- Based on emulator
- Extended by energy model
- Emulation of ALL hardware components
- Original code in application and OS



Evaluating target scenarios with the energy model

Energy Analysis:
- TinyOS components
  - Power Management
  - Communication
- TinyOS applications
- Node lifetime
- Network lifetime

➡ Predicted Energy Consumption by time



Eprof (Pathak et al.)

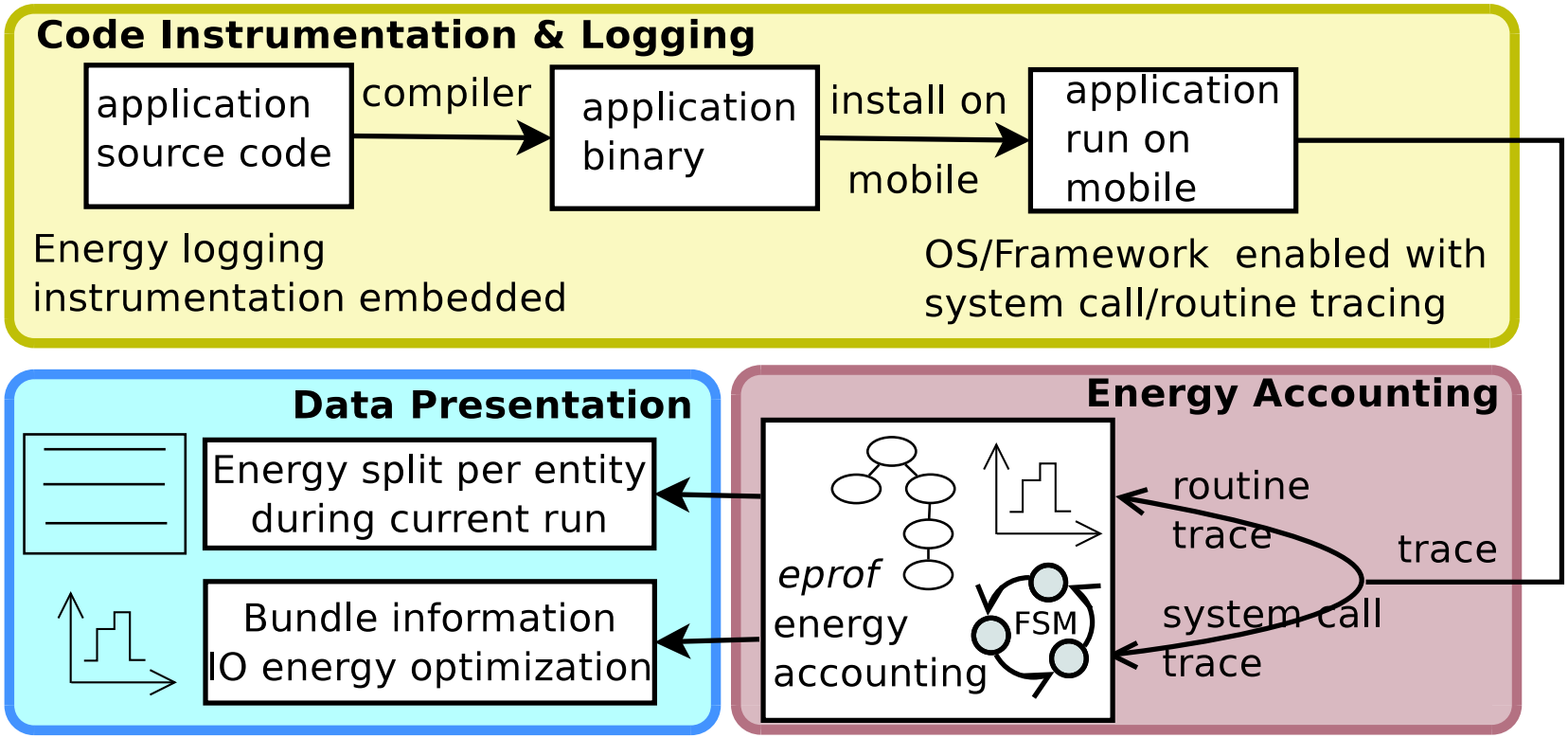


Abbildung: Überblick über Eprof von Pathak et al. (2012). Quelle: [1]

Fazit

- Verschiedene Umsetzungen für verschiedene Anforderungen
- Direkter Vergleich?
- Zusatzaufwand des Profilings gering halten
  $\rightarrow$ Beeinflussung der Messergebnisse?

Überblick

Entwurfsaspekte beim Profiling

Beispiele für Profiler-Umsetzungen

Zusammenfassung

Zusammenfassung

- Entwurfsaspekte für Energie-Profiler
- Asynchroner Energieverbrauch
  - kein einfaches „Verstecken“ des Problems möglich
  - Bewusstsein dafür beim Nutzer des Energie-Profilers schaffen!
- Verschiedene Profiler-Umsetzungen für verschiedene Anforderungen möglich

Fragen?

Quellen

A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, (New York, NY, USA), pp. 29–42, ACM, 2012.

T. Hönig, C. Eibel, R. Kapitza, and W. Schröder-Preikschat, "SEEP: exploiting symbolic execution for energy-aware programming," *SIGOPS Oper. Syst. Rev.*, vol. 45, pp. 58–62, Jan. 2012.

J. Flinn and M. Satyanarayanan, "Powerscope: a tool for profiling the energy usage of mobile applications," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pp. 2–10, 1999.

O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate prediction of power consumption in sensor networks," in *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, EmNets '05, (Washington, DC, USA), pp. 37–44, IEEE Computer Society, 2005.