Laufzeitumgebungen und Software-Frameworks

Seminar „Ausgewählte Kapitel der Systemsoftwaretechnik: Energiegewahre Systemsoftware“

Jeremias Isnardy

11. Juli 2013

Motivation

- Verbrauch von Energie ist ein Hauptaugenmerk für Entwickler
- Im Seminar bisher behandelt:
  - Hardware-Mechanismen zur Optimierung des Energieverbrauchs
  - externe Werkzeuge für Analyse bestehender Programme (Energie-Profiler)
- Aber: Auf Programmier-Ebene noch viel ungenutztes Potential
- $\Rightarrow$ Vorstellung von Methoden für die Sprach- und Laufzeitumgebung

Inhaltsangabe

- Eon – Programmiersprache und Laufzeit-System
  - Konzept
  - Laufzeit-System
  - Untersuchungen
- EnerJ – Ungefähre Daten-Typen
  - Konzept
  - Hardware-Umsetzung
  - Untersuchungen
- Zusammenfassung

Eon – Programmiersprache und Laufzeit-System

- Für beständige Systeme (perpetual systems) entwickelt
- Verschiedene Energiezustände definierbar und somit die Möglichkeit für Programmierer verschiedene Szenarien zu implementieren.
- Laufzeit-System sorgt für möglichst exakte und reibungslose Umsetzung der gewünschten Implementierung

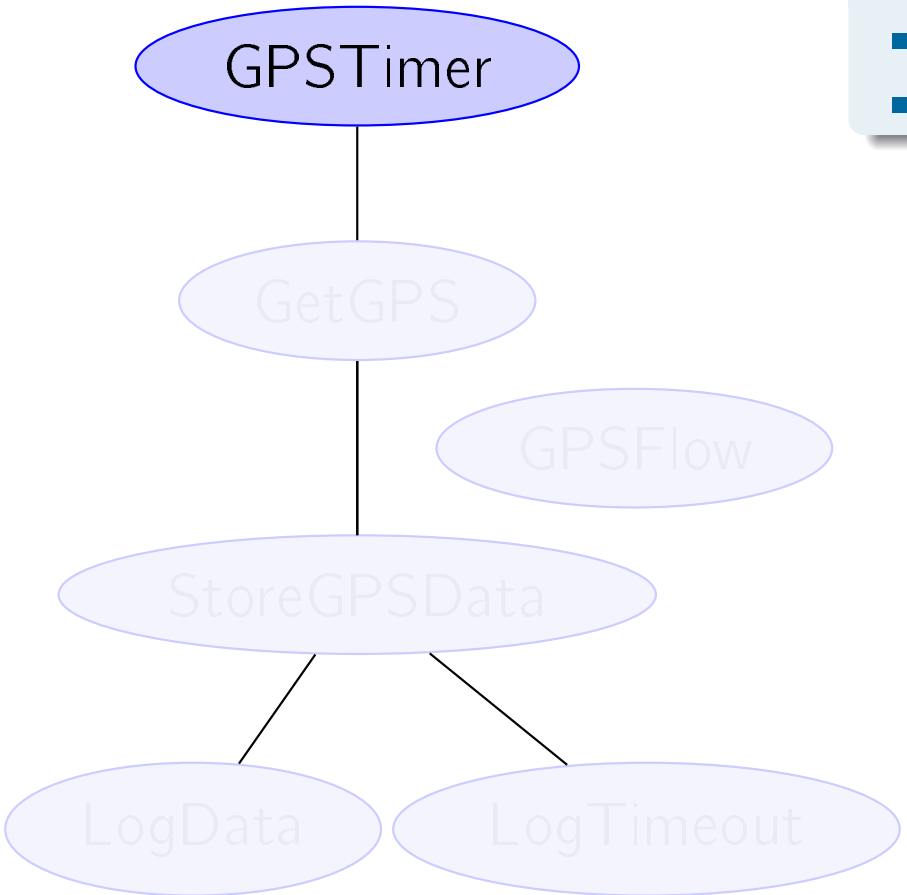
Beständige Systeme

- Systeme, die theoretisch unendlich lange laufen
- Energiegewinnung aus unmittelbarer Umgebung (Energy-Harvesting)
- Energiequellen sind allerdings unzuverlässig und Zufuhr daher schwankend
- Je nach Anwendung auch Energieverbrauch unbeständig
- Programme für diese Systeme müssen flexibel reagieren können

Konzept

- Definition verschiedener Energiezustände möglich
- Programmierer weist Teile des Codes verschiedene Prioritäten zu
- Je nach Energiezustand werden Teile des Codes dementsprechend seltener oder gar nicht ausgeführt
- Zuordnung der Prioritäten sehr sensibel, Programmierer muss sich über Auswirkungen im Klaren sein

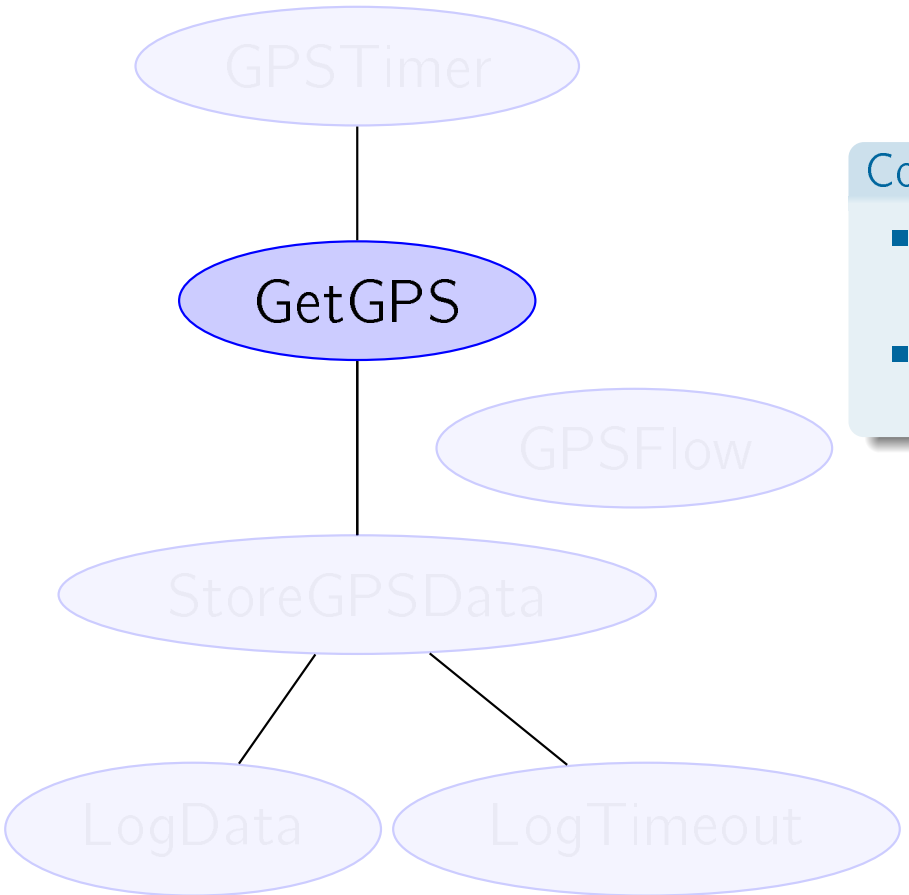
Syntax



Source-Node

- Übergeordnete Datenquelle
- Stellt benötigte Daten zur Verfügung

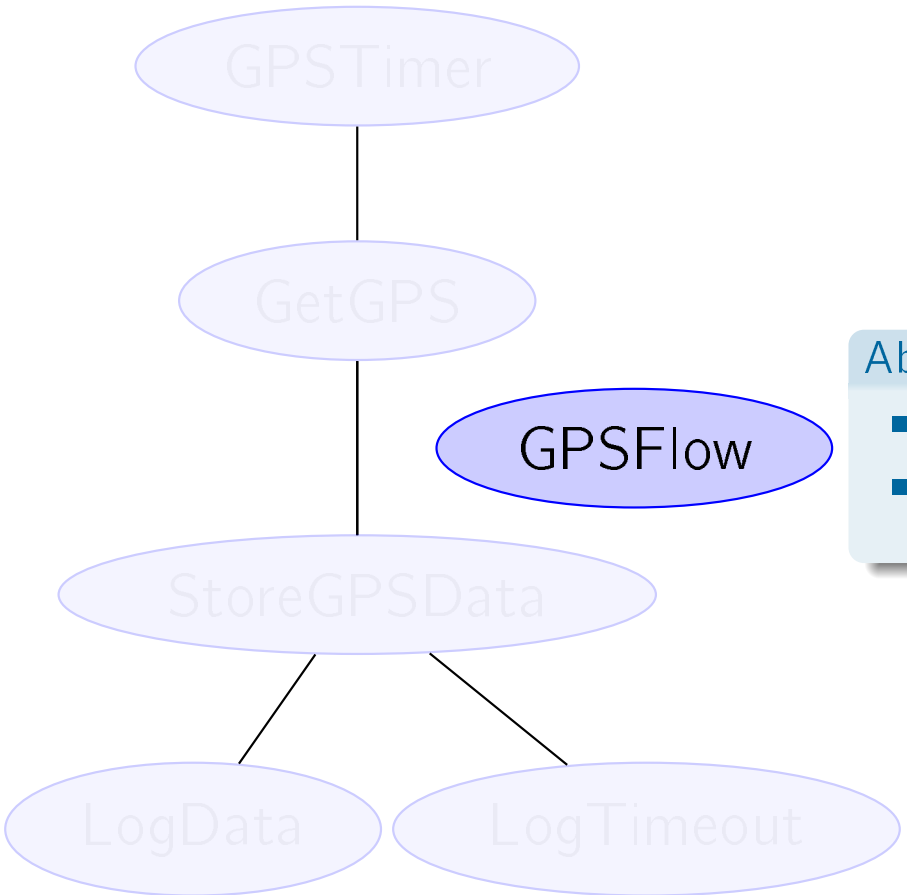
Syntax



Concrete-Node

- Verweist auf Funktionen, die in C/nesC geschrieben sind
- Bekommt Daten von Source-Node, erzeugt Ausgabedaten

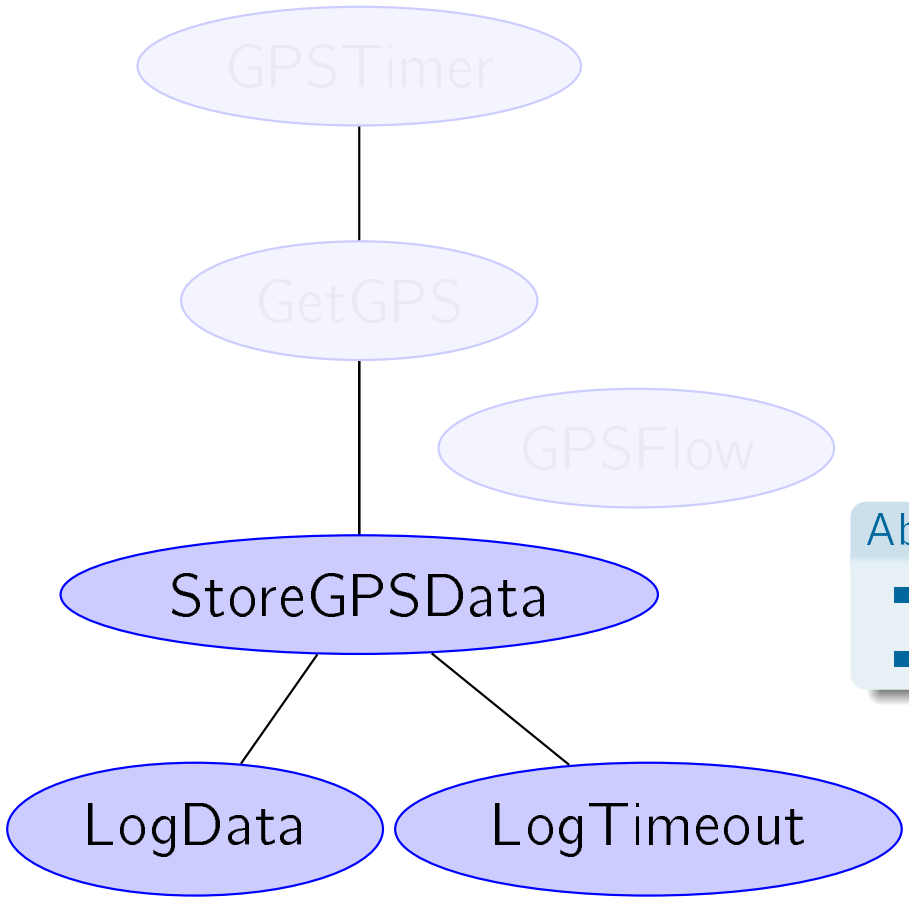
Syntax



Abstract-Node

- Steuert den Programmfluss
- Ruft Concrete-Nodes oder andere Abstract-Nodes auf

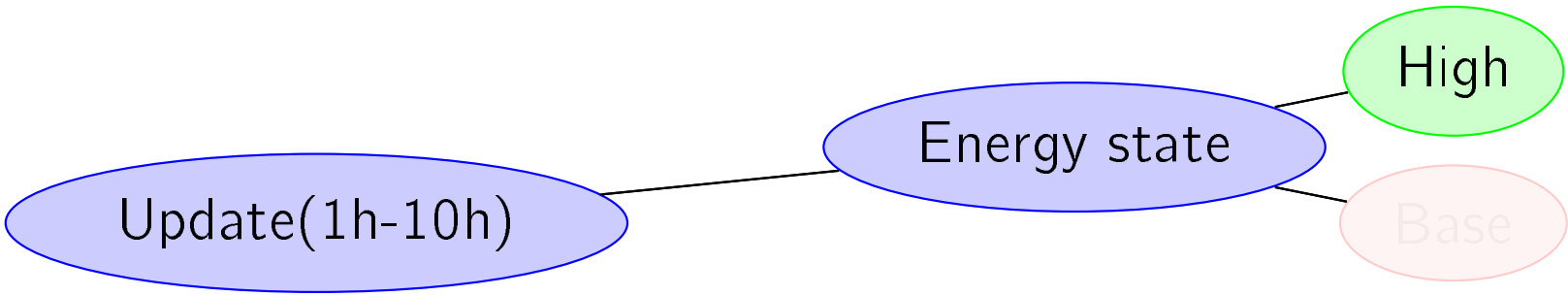
Syntax

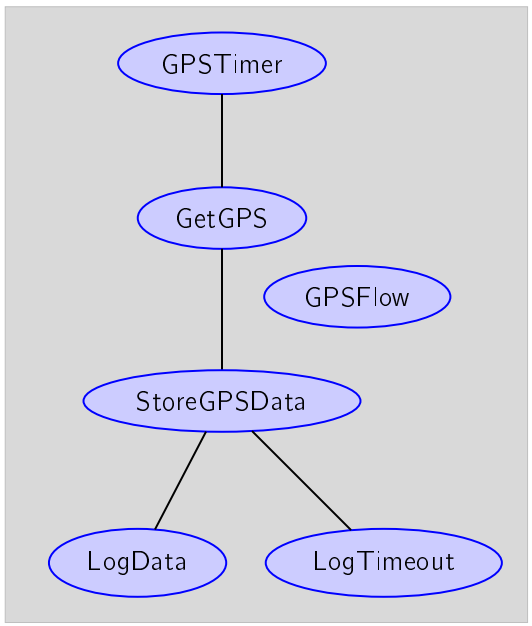


Abstract-Node: Conditional-Flow

- Gesteuert durch Predicated-Types
- Hier: Einfluss des Programmierers

Syntax

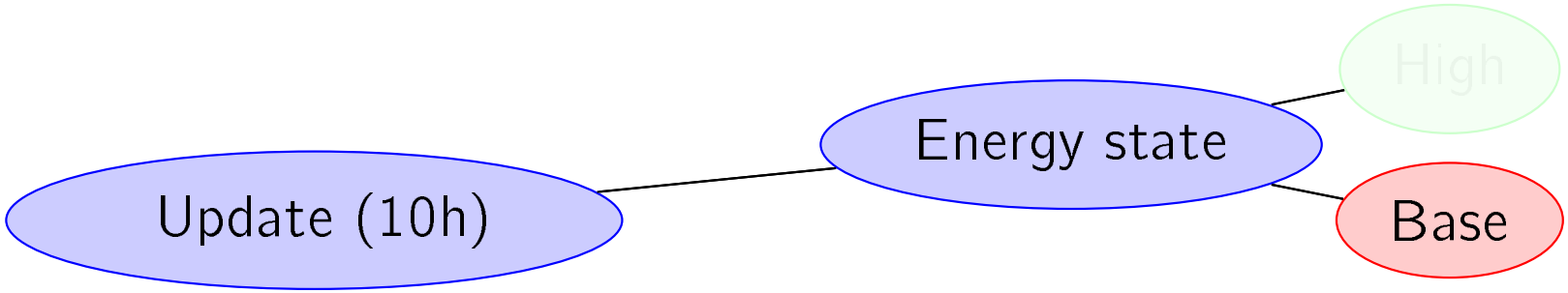


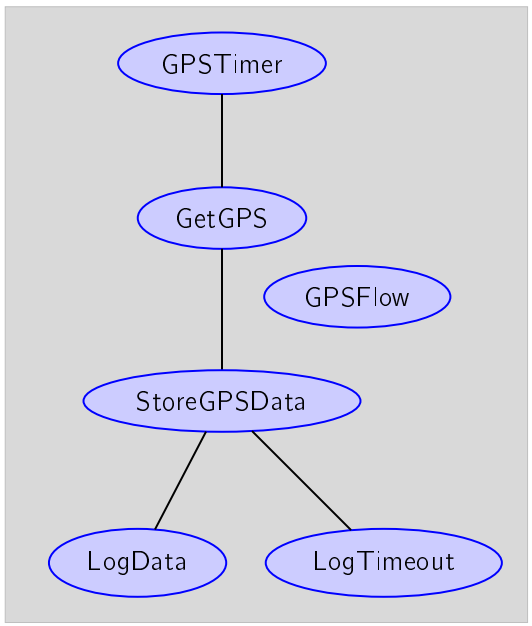


Energy State

- Definition zusätzlicher Energiezustände (hier: high)
- Übergeordnete Steuerung des Programmflusses

Syntax





Adaptive-Timer

- Definition von Zeitintervallen
- Je nach Energiezustand wird Zeit aus entsprechendem Intervall gewählt
- high: Updatezeit zwischen einer und zehn Stunden
- base: Updatezeit beträgt zehn Stunden

Laufzeit-System

- Wahl des besten Energiezustandes Aufgabe des Laufzeit-Systems
- Algorithmus sucht optimalen Zustand zunächst für kurze, dann für längere Intervalle
- Nötige Informationen:
  - bisheriger Energieverbrauch und Energiegewinnung wird über einen exponentiell geglätteten Mittelwert bestimmt
  - zukünftiger Energieverbrauch und Energiegewinnung wird über Algorithmen mit der Annahme bestimmt, dass diese in etwa dem vergangenen Verhalten entsprechen

Experiment

- GPS-Verfolgung von Autos
  - fünf Autos wurden mit GPS-Geräten ausgestattet
  - Energieversorgung durch Solarenergie
  - Wetter unbeständig, dementsprechend auch Energiezufuhr
  - zweiwöchige Datensammlung
- Auswertung
  - Anzahl der GPS-Updates und Ladezustand der Batterie
  - Daten wurden in Simulation auf drei Monate hochgerechnet
  - Vergleich mit anderen Strategien für die Rate der GPS-Updates:
    - Conservative: niedrigste Rate aller Proben
    - Greedy: größte Rate aller Proben
    - Best-Static: beste mögliche Rate für jede Probe
    - Eon (Oracle): Eon mit idealer Vorhersage des Wetters

Ergebnisse

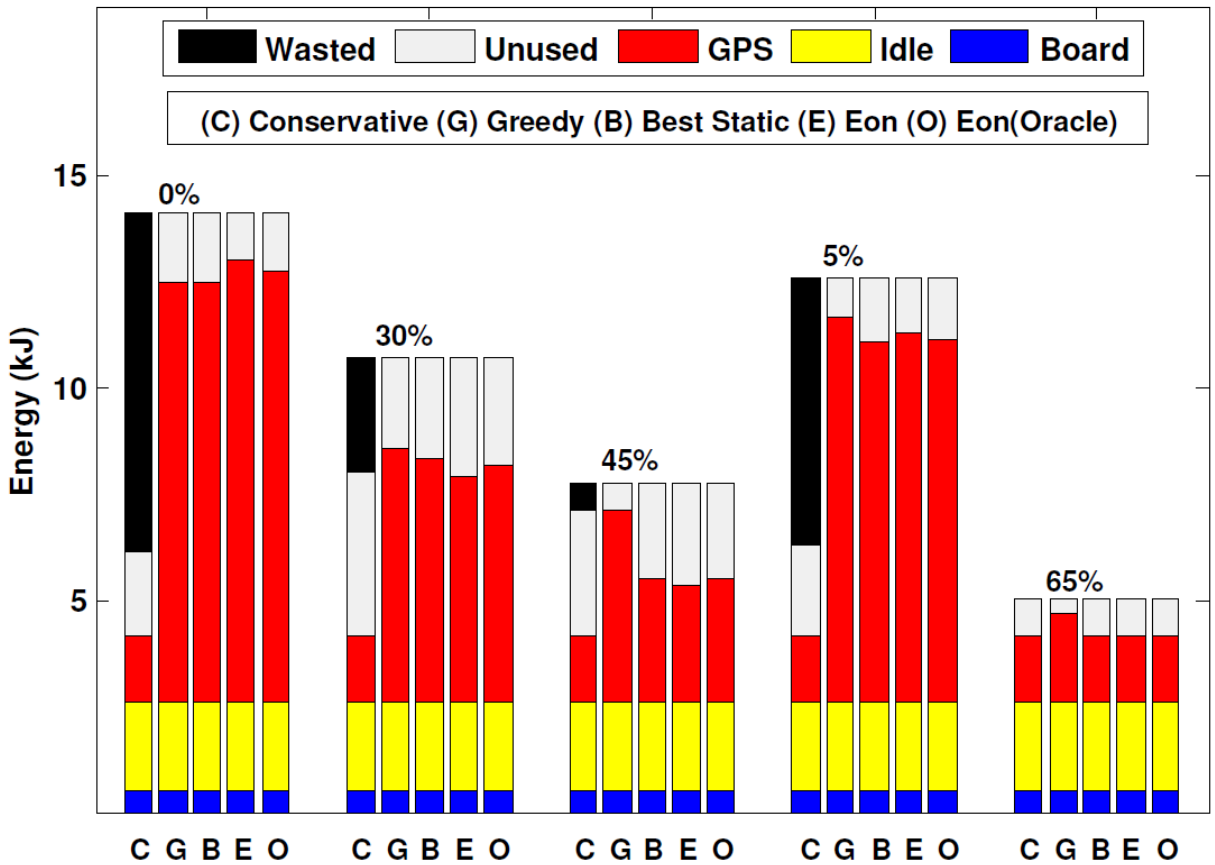


Abbildung: Vergleich des Energieverbrauchs durch verschiedene Teile des Systems, für jede Probe mit jeder Strategie [2]

Weitere Beobachtungen

- Benutzerfreundlicher Einstieg, leicht zu erlernen
- Energieverbrauch für Bestimmung des Energiezustandes verhältnismäßig gering
- Einfluss des Schätzungsfehler des Energieverbrauchs:
  - Große Batterie-Kapazität (>150 mAh) kann mögliche Fehler abfangen
  - Kleine Batterie-Kapazität kann zur Ausfallzeit dieser führen

EnerJ – Ungefähre Daten-Typen

- Viele Programme oder Teile in Programmen können Ungenauigkeiten tolerieren
- Für diese Bereiche genügen ungefähre Berechnungen
- EnerJ: Erweiterung von Java um ungefähre Daten-Typen
  - weniger Speicherplatz nötig
  - weniger Aufwand für Berechnungen
  - $\Rightarrow$ weniger Energieverbrauch

Typen-System

- `@Approx` vor einem Datentyp
- `@Approximable` vor einer Klassen-Definition
- `@Context` für Typen in der Klasse, die von der Klasseninstanz abhängen
- `_APPROX` als Suffix für Methoden

  Zu beachten: Datenfluss nur von präzisen zu ungefähren Daten zulässig

Typen-System

```
@Approximable class IntPair {
     @Context int x;
     @Context int y;
     @Approx int numAdditions = 0;
     void addToBoth(@Context int amount) {
          x += amount;
          y += amount;
          numAdditions++;
     }
}
```

```
@Approximable class FloatSet {
  @Context float[] nums = ...;

  float mean() {
     float total = 0.0f;
     for (int i = 0; i < nums.length; ++i)
          total += nums[i];
     return total / nums.length;
 }

 @Approx float mean_APPROX() {
     @Approx float total = 0.0f;
     for (int i = 0; i < nums.length; i += 2)
          total += nums[i];
     return 2 * total / nums.length;
 }
}
```

Hardware-Umsetzung

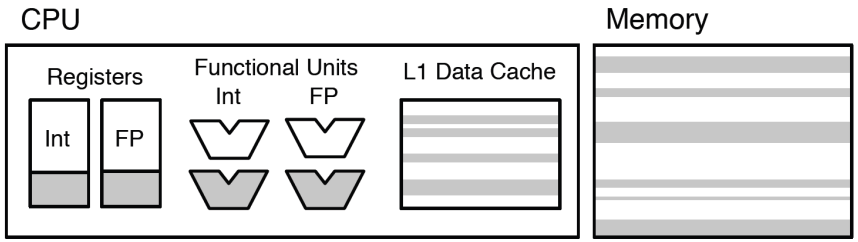


Abbildung: Hardware-Modell für Nutzung von ungefähren Daten-Typen [1]

- Register, Cache und DRAM
  - SRAM-Speicher: Senkung der Versorgungsspannung
  - Senkung der Aktualisierungsrate
- Steuereinheit
  - Senkung der Versorgungsspannung
  - Kürzung der Mantisse

Experiment

- Drei Strategien für Hardwareparameter (Mild, Medium, Aggressive)
- Angepasste Programme als Benchmarks:

Application	Lines of code	Prop. FP	Total decls.	Annot. decls.	Endorse-ments
FFT	168	38,2 %	85	33%	2
SOR	36	55,2 %	28	25%	0
MonteCarlo	59	22,9 %	15	20%	1
SMM	38	39,7 %	29	14%	0
LU	283	31,4 %	150	23%	3
ZXing	26171	1,7 %	11506	4%	247
jMonkeyEng.	5962	44,2 %	2104	19%	63
ImageJ	156	0,0 %	118	34%	18
Raytracer	174	68,4 %	92	33%	10

Tabelle: Als Benchmarks genutzte Anwendungen [1]

Auswertung

Folgende Annahmen wurden getroffen:

- Kein Einfluss bei Wechsel der Hardwareeinstellungen
- Nur drei Komponenten beeinflussen Energieverbrauch
  - Befehlsausführung: Jeder Operation bekommt eine abstrakte Energieeinheit zugeordnet
  - SRAM: Speicherung im SRAM und Befehle die darauf zugreifen machen 35% der verbrauchten Energie auf CPU-Ebene aus
  - DRAM: 45% des gesamten Energieverbrauchs gegenüber 55% der CPU

$\Rightarrow$ Resultate fallen optimistisch aus.

Ergebnisse

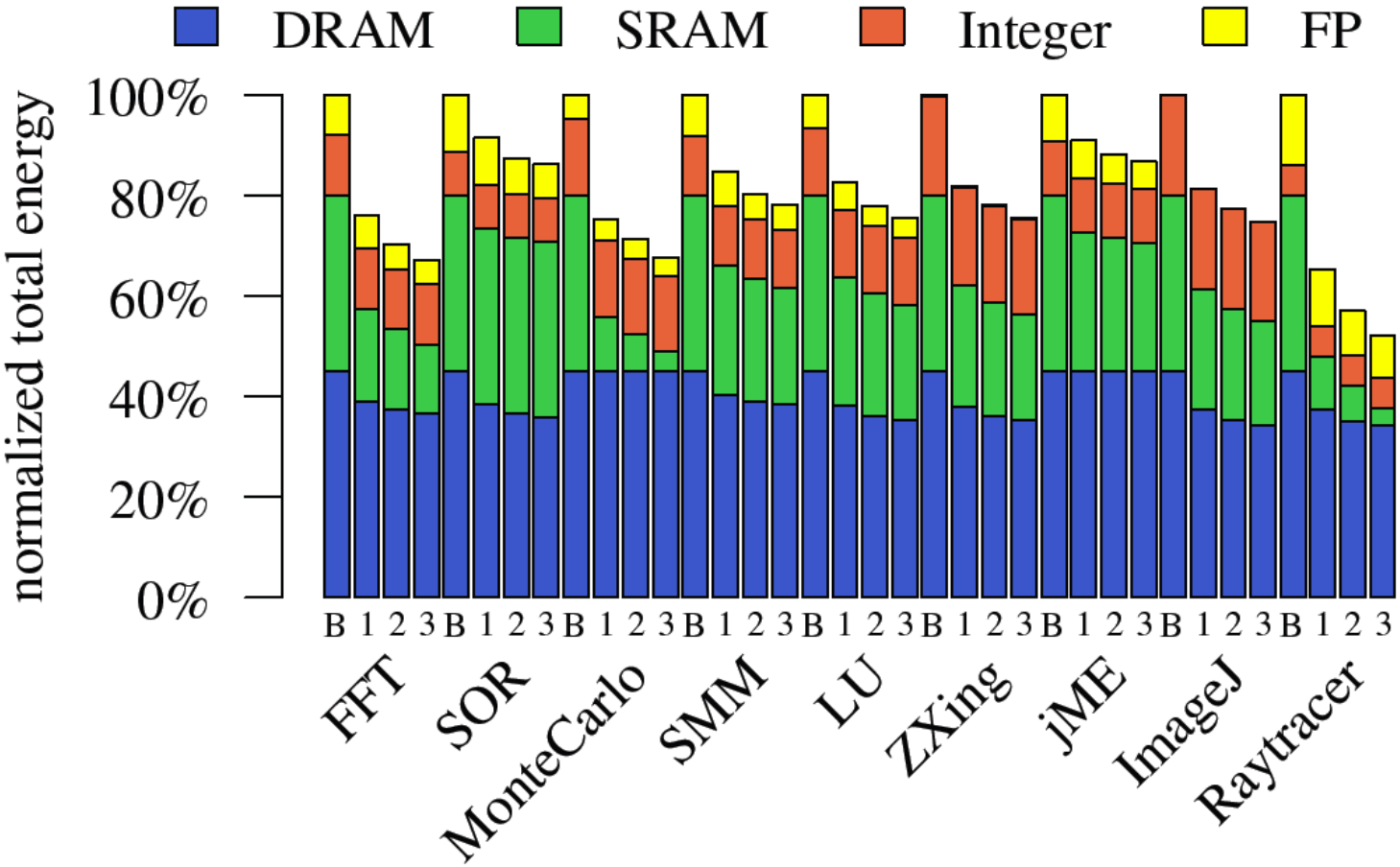


Abbildung: Geschätzter Energieverbrauch für jede Benchmark [1]

Ergebnisse

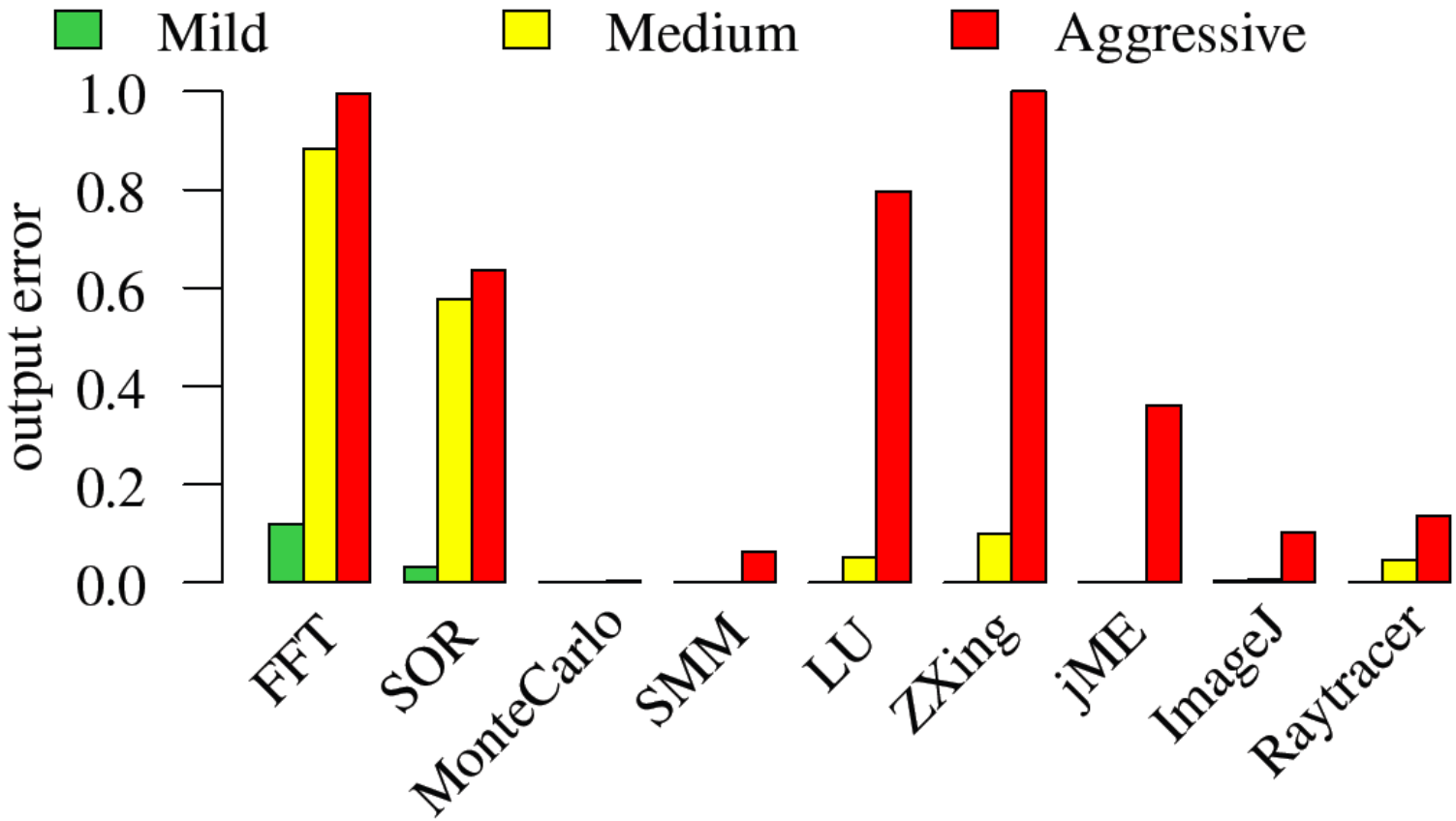


Abbildung: Entstandene Fehler bei den Benchmarks, abhängig von den genutzten Strategien [1]

Zusammenfassung

- Eon
  - Für beständige System entwickelt und geeignet
  - Leicht anzuwenden
  - Von Programmlogik getrennt
  - Aber: Energieersparnis ausschließlich auf Kosten der Systemleistung
- EnerJ
  - Spart Energie an Stellen, wo sie sonst ineffizient verschwendet wird
  - Hardware muss exakt angepasst werden
  - Weitsicht des Programmierers notwendig
  - Genauere Untersuchungen der effektiven Energieersparnis erforderlich

Zusammenfassung

Erweiterungen von Programmiersprachen und speziellen Ausführungsumgebungen guter Ansatz.

Aber: Energieersparnis ist nicht alles

- Lohnt sich der Mehraufwand für den Programmierer?
- Nutzen/Aufwand-Verhältnis angemessen?
- Nachhaltigkeit des genutzten Konzepts?
- Wird die Hardware-Portabilität des Programms beeinflusst?
- Integration in bereits bestehende Softwareprojekte möglich?

FRAGEN?

Vielen Dank
Für Ihre Aufmerksamkeit


Sampson, A., Dietl, W., Fortuna, E., Gnanapragasam, D., Ceze, L., and Grossman, D.
EnerJ: Approximate Data Types for Safe and General Low-Power Computation.
In *Proceedings of Programming Language Design and Implementation (PLDI '11), California* (June 2011).


Sorber, J., Kostadinov, A., Garber, M., Brennan, M., Corner, M. D., and Berger, E. D.
Eon: A language and runtime system for perpetual systems.
In *Proceedings of The Fifth International ACM Conference on Embedded Networked Sensor Systems (SenSys '07), Syndey* (Nov. 2007).

Ergebnisse

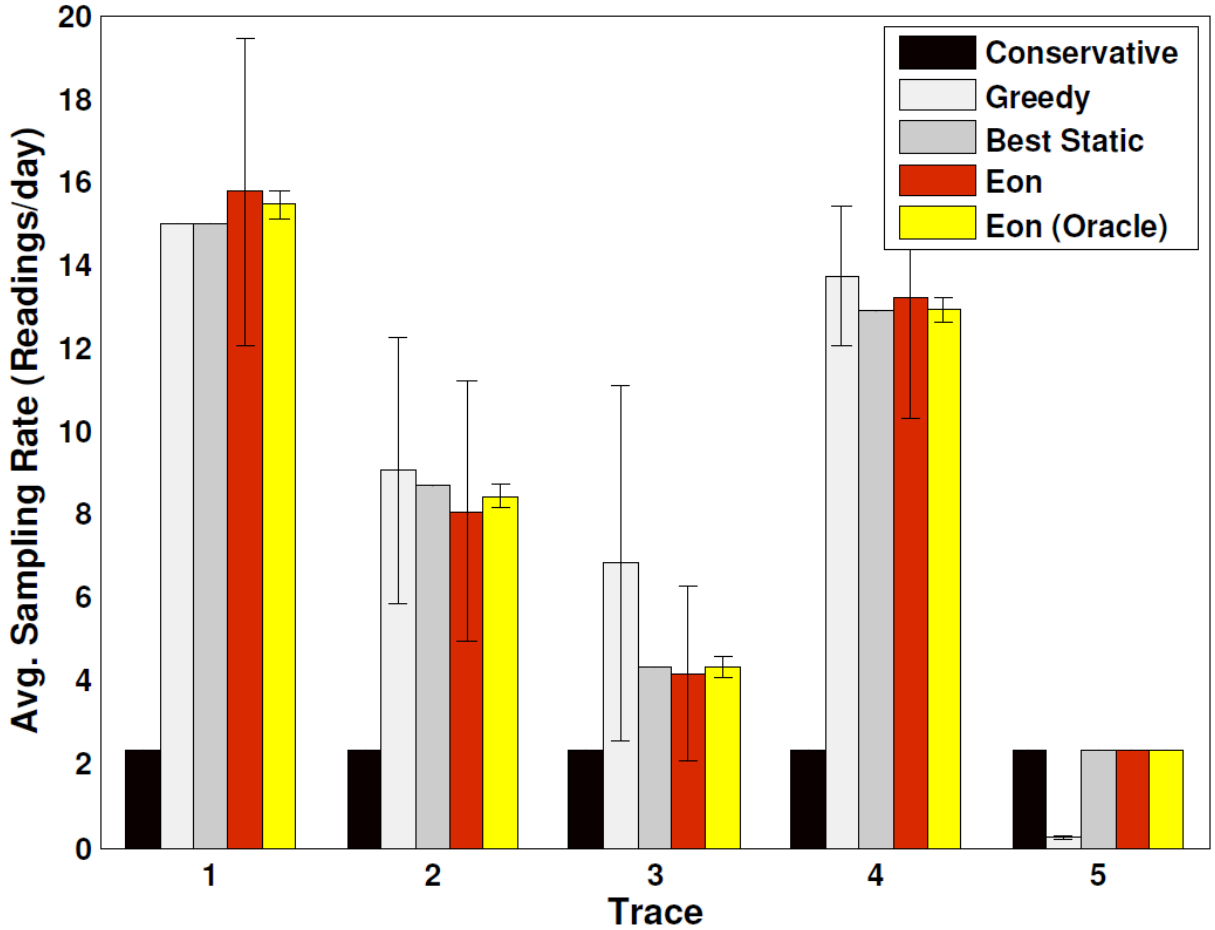


Abbildung: Durchschnittliche Aktualisierungsrate der GPS-Koordinaten verschiedener Strategien für jede Probe [2]