Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

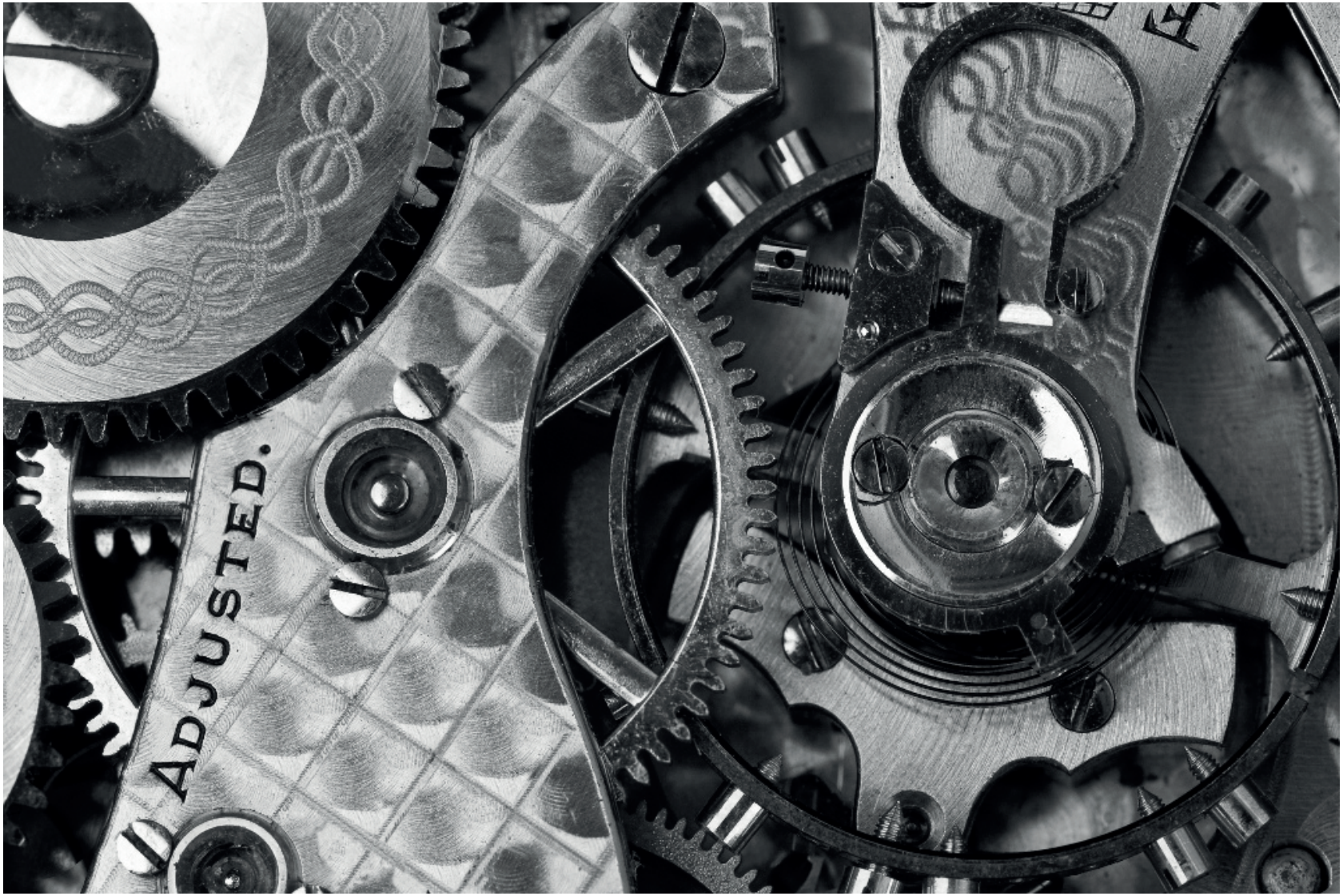
XII. Nachlese

Wolfgang Schröder-Preikschat

7. Juli 2014

Gliederung

Adressraumverwaltung



Verfeinerung/Vertiefung von Betriebssysteme [1, 2]

Adressräume (von Programmen/Prozessen)

- tren|nen: in eine räumliche Distanz voneinander bringen
  - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
  - unterstützt durch Dienstprogramme (*utility program*)
    - Kompilierer, Assemblierer, Binder, Lader
  - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- zu|grei|fen: nach etwas greifen und es festhalten bzw. an sich nehmen
  - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
  - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
  - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- schüt|zen: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
  - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
  - Immunität einerseits und Isolation andererseits
  - Eindrang bzw. Ausbruch von Prozessen verhindern

Verfeinerung/Vertiefung von Betriebssysteme [1, 2]

Adressräume (von Programmen/Prozessen)

- tren|nen: in eine räumliche Distanz voneinander bringen
  - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
  - unterstützt durch Dienstprogramme (*utility program*)
    - Kompilierer, Assemblierer, Binder, Lader
  - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- zu|grei|fen: nach etwas greifen und es festhalten bzw. an sich nehmen
  - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
  - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
  - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- schüt|zen: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
  - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
  - Immunität einerseits und Isolation andererseits
  - Eindrang bzw. Ausbruch von Prozessen verhindern

Verfeinerung/Vertiefung von Betriebssysteme [1, 2]

Adressräume (von Programmen/Prozessen)

- tren|nen: in eine räumliche Distanz voneinander bringen
  - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
  - unterstützt durch Dienstprogramme (*utility program*)
    - Kompilierer, Assemblierer, Binder, Lader
  - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- zu|grei|fen: nach etwas greifen und es festhalten bzw. an sich nehmen
  - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
  - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
  - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- schüt|zen: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
  - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
  - Immunität einerseits und Isolation andererseits
  - Eindrang bzw. Ausbruch von Prozessen verhindern

Verfeinerung/Vertiefung von Betriebssysteme [1, 2]

Adressräume (von Programmen/Prozessen)

- tren|nen: in eine räumliche Distanz voneinander bringen
  - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
  - unterstützt durch Dienstprogramme (*utility program*)
    - Kompilierer, Assemblierer, Binder, Lader
  - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- zu|grei|fen: nach etwas greifen und es festhalten bzw. an sich nehmen
  - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
  - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
  - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- schüt|zen: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
  - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
  - Immunität einerseits und Isolation andererseits
  - Eindrang bzw. Ausbruch von Prozessen verhindern

Verfeinerung/Vertiefung von Betriebssysteme [1, 2]

Adressräume (von Programmen/Prozessen)

- tren|nen: in eine räumliche Distanz voneinander bringen
  - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
  - unterstützt durch Dienstprogramme (*utility program*)
    - Kompilierer, Assemblierer, Binder, Lader
  - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- zu|grei|fen: nach etwas greifen und es festhalten bzw. an sich nehmen
  - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
  - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
  - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- schüt|zen: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
  - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
  - Immunität einerseits und Isolation andererseits
  - Eindrang bzw. Ausbruch von Prozessen verhindern

↪ ergänzend: softwarebasiert, durch typsichere Programmiersprachen

Gliederung

Forschungsschwerpunkte am Lehrstuhl

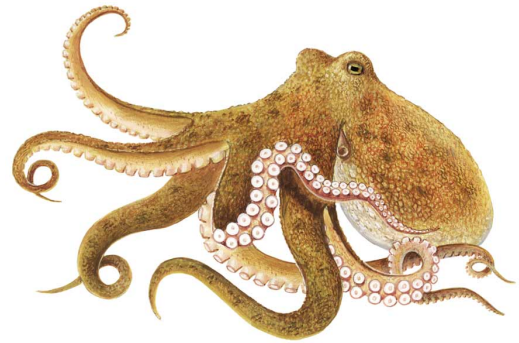
- **Komponierbarkeit** und **Konfigurierbarkeit**
  - anwendungsorientierte (variantenreiche, typsichere) Systemsoftware
- **Sparsamkeit**
  - ressourcen-gewahrer Betrieb von Rechensystemen
- **Zuverlässigkeit**
  - Betriebsmittel schonende Fehler- und Einbruchstoleranz
- **Rechtzeitigkeit**
  - Migrationspfade zwischen zeit- und ereignisgesteuerten Echtzeitsystemen
- **Spezialisierbarkeit**
  - dedizierte Betriebssysteme: integriert, adaptiv, parallel
- **Gleichzeitigkeit**
  - Koordination der Kooperation und Konkurrenz zwischen Prozessen

Forschungsschwerpunkte am Lehrstuhl

- **Komponierbarkeit** und **Konfigurierbarkeit**
  - anwendungsorientierte (variantenreiche, typsichere) Systemsoftware
- **Sparsamkeit**
  - ressourcen-gewahrer Betrieb von Rechensystemen
- **Zuverlässigkeit**
  - Betriebsmittel schonende Fehler- und Einbruchstoleranz
- **Rechtzeitigkeit**
  - Migrationspfade zwischen zeit- und ereignisgesteuerten Echtzeitsystemen
- **Spezialisierbarkeit**
  - dedizierte Betriebssysteme: integriert, adaptiv, parallel
- **Gleichzeitigkeit**
  - Koordination der Kooperation und Konkurrenz zwischen Prozessen

↪ Prozessadressräume sind mehr oder weniger querschneidend dazu

Laufzeitunterstützungssystem für invasives Rechnen

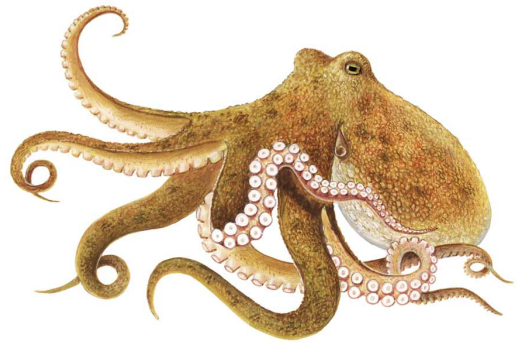


[1] http://univis.uni-erlangen.de → Forschungsprojekte → iRTSS

Laufzeitunterstützungssystem für invasives Rechnen

Octo

- der Bezeichnung eines Wesens entnommen, das:
  - i hoch parallel in seinen Aktionen ist und
  - ii sich sehr gut an seine Umgebung anpassen kann
- der Krake (Ordnung *Octopoda*)
  - kann kraft seiner (acht) Tentakel parallel agieren
  - vermag sich durch Farbänderung anzupassen und
  - verfügt über ein hoch entwickeltes Nevensystem
    - um sich auf dynamische Umgebungsbedingungen und -einflüsse einzustellen



POS

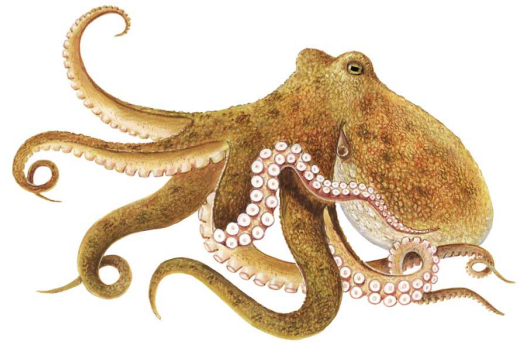
- Abk. für (engl.) *Parallel Operating System*
  - ein Betriebssystem, das nicht bloß parallele Prozesse unterstützt
  - sondern dabei selbst inhärent parallel arbeitet

[1] `http://univis.uni-erlangen.de` → Forschungsprojekte → iRTSS

Laufzeitunterstützungssystem für invasives Rechnen

Octo

- der Bezeichnung eines Wesens entnommen, das:
  - i hoch parallel in seinen Aktionen ist und
  - ii sich sehr gut an seine Umgebung anpassen kann
- der Krake (Ordnung *Octopoda*)
  - kann kraft seiner (acht) Tentakel parallel agieren
  - vermag sich durch Farbänderung anzupassen und
  - verfügt über ein hoch entwickeltes Nevensystem
    - um sich auf dynamische Umgebungsbedingungen und -einflüsse einzustellen



POS

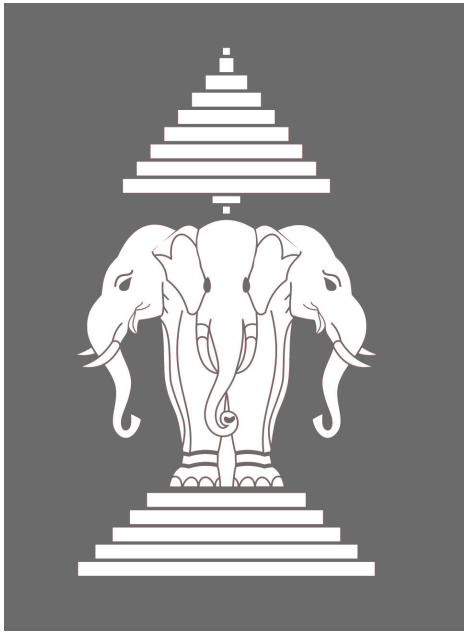
- Abk. für (engl.) *Parallel Operating System*
  - ein Betriebssystem, das nicht bloß parallele Prozesse unterstützt
  - sondern dabei selbst inhärent parallel arbeitet

↪ 64-Kerner (transaktionaler Speicher, Haswell) für die 2. Phase

[1] `http://univis.uni-erlangen.de` → Forschungsprojekte → iRTSS

Latency Awareness in Operating Systems

Latenzgewahrheit in Betriebssystemen für massiv-parallele CPUs

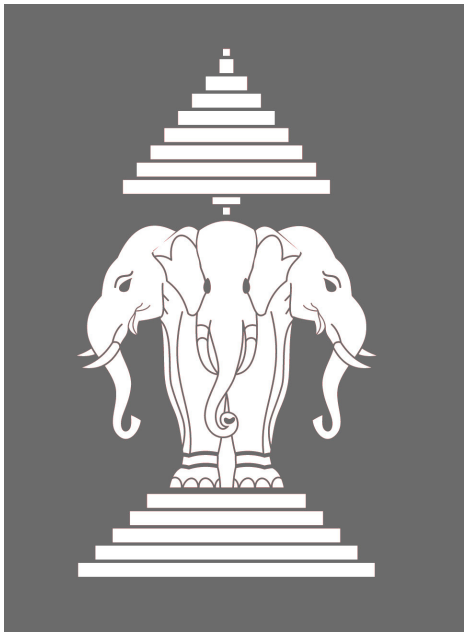


[2] http://univis.uni-erlangen.de → Forschungsprojekte → LAOS

Latency Awareness in Operating Systems

Latenzgewahrheit in Betriebssystemen für massiv-parallele CPUs

- **Latenzvorbeugung**
  - domänenspezifische Entwurfsmuster
  - sperr- und wartefreie Synchronisation
- **Latenzvermeidung**
  - Interferenzschutz
  - Eindämmung von Wettstreitigkeiten
- **Latenzverbergung**
  - asynchrone Prozedurfern-/Systemaufrufe
  - Kerne für Betriebssysteme nutzen

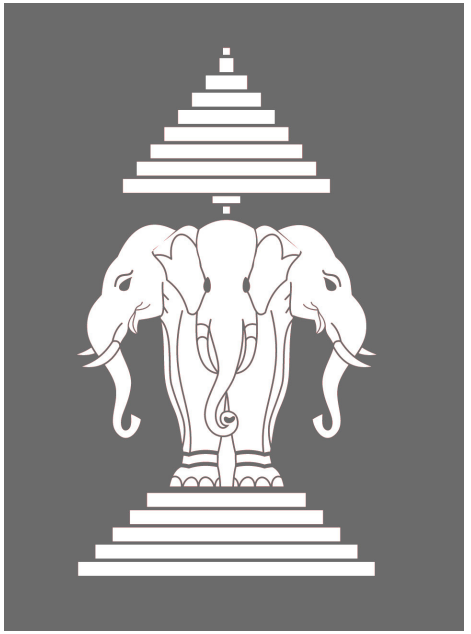


[2] `http://univis.uni-erlangen.de` → Forschungsprojekte → LAOS

Latency Awareness in Operating Systems

Latenzgewahrheit in Betriebssystemen für massiv-parallele CPUs

- **Latenzvorbeugung**
  - domänenspezifische Entwurfsmuster
  - sperr- und wartefreie Synchronisation
- **Latenzvermeidung**
  - Interferenzschutz
  - Eindämmung von Wettstreitigkeiten
- **Latenzverbergung**
  - asynchrone Prozedurfern-/Systemaufrufe
  - Kerne für Betriebssysteme nutzen



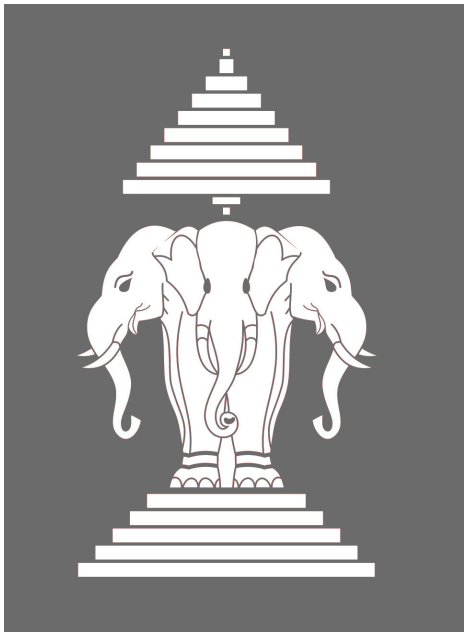
- Experimente mit unterschiedlichen **Betriebssystemarchitekturen**
  - Entwicklung eigener prozess- und ereignisbasierte Betriebssystemkerne
  - Übertragung mancher Konzepte und Techniken in Linux

[2] `http://univis.uni-erlangen.de` → Forschungsprojekte → LAOS

Latency Awareness in Operating Systems

Latenzgewahrheit in Betriebssystemen für massiv-parallele CPUs

- **Latenzvorbeugung**
  - domänenspezifische Entwurfsmuster
  - sperr- und wartefreie Synchronisation
- **Latenzvermeidung**
  - Interferenzschutz
  - Eindämmung von Wettstreitigkeiten
- **Latenzverbergung**
  - asynchrone Prozedurfern-/Systemaufrufe
  - Kerne für Betriebssysteme nutzen



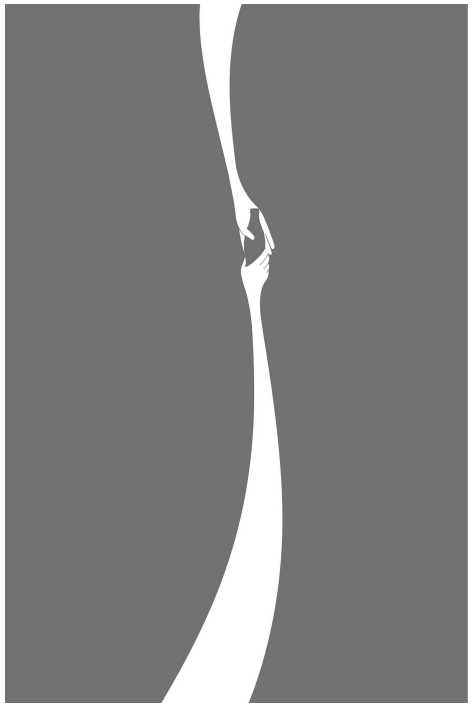
- Experimente mit unterschiedlichen **Betriebssystemarchitekturen**
  - Entwicklung eigener prozess- und ereignisbasierte Betriebssystemkerne
  - Übertragung mancher Konzepte und Techniken in Linux

↪ 48-kerniges Rechensystem („Bigbox“)

[2] `http://univis.uni-erlangen.de` → Forschungsprojekte → LAOS

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

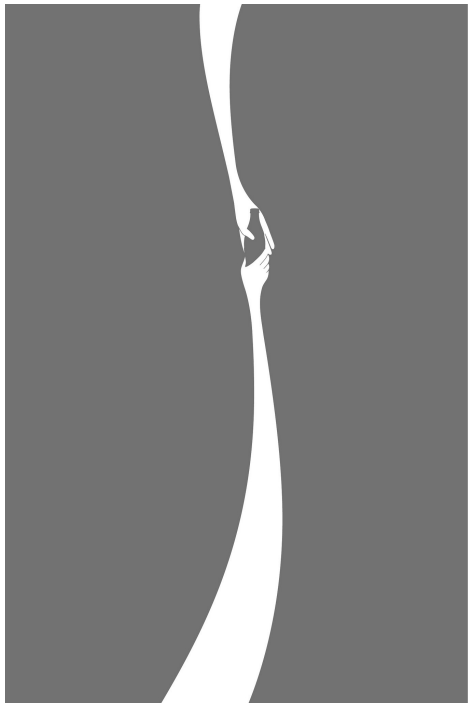


[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

- **ereignisbasierter Minimalkern**
  - zwischenspeichergewahrer Speicherabdruck
  - „überfaden“ (*hyper-threading*) latenter Aktionen

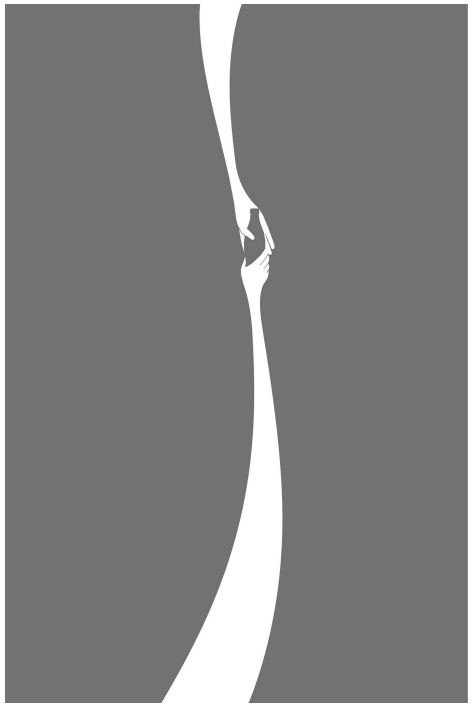


[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

- **ereignisbasierter Minimalkern**
  - zwischenspeichergewahrer Speicherabdruck
  - „überfaden“ (*hyper-threading*) latenter Aktionen
- federgewichtige **Einigungsprotokolle**
  - kernübergreifende Synchronisation
  - Familie von Konsistenzkernen



[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

- **ereignisbasierter Minimalkern**
  - zwischenspeichergewahrer Speicherabdruck
  - „überfaden“ (*hyper-threading*) latenter Aktionen
- federgewichtige **Einigungsprotokolle**
  - kernübergreifende Synchronisation
  - Familie von Konsistenzkernen
- **problemorientierte Konsistenzmaschinen**
  - sequentielle, Eintritts- und Freigabekonsistenz
  - funktionale Hierarchie von Konsistenzdomänen
  - Speicherdomänen für NUMA-Architekturen

[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

- **ereignisbasierter Minimalkern**
  - zwischenspeichergewahrer Speicherabdruck
  - „überfaden“ (*hyper-threading*) latenter Aktionen
- federgewichtige **Einigungsprotokolle**
  - kernübergreifende Synchronisation
  - Familie von Konsistenzkernen
- **problemorientierte Konsistenzmaschinen**
  - sequentielle, Eintritts- und Freigabekonsistenz
  - funktionale Hierarchie von Konsistenzdomänen
  - Speicherdomänen für NUMA-Architekturen
- Auslegungen für unterschiedliche **Prozessorarchitekturen**
  - partiell bzw. total, {in,}kohärenter gemeinsamer Speicher

[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Coherency Kernel

Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren

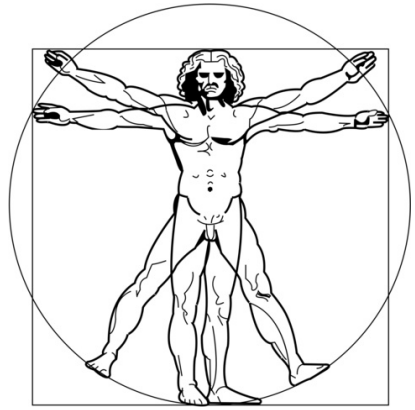
- **ereignisbasierter Minimalkern**
  - zwischenspeichergewahrer Speicherabdruck
  - „überfaden“ (*hyper-threading*) latenter Aktionen
- federgewichtige **Einigungsprotokolle**
  - kernübergreifende Synchronisation
  - Familie von Konsistenzkernen
- **problemorientierte Konsistenzmaschinen**
  - sequentielle, Eintritts- und Freigabekonsistenz
  - funktionale Hierarchie von Konsistenzdomänen
  - Speicherdomänen für NUMA-Architekturen
- Auslegungen für unterschiedliche **Prozessorarchitekturen**
  - partiell bzw. total, {in,}kohärenter gemeinsamer Speicher

↪ 64-Kerner (transaktionaler Speicher), ggf. zusätzlich Intel SCC

[3] `http://univis.uni-erlangen.de` → Forschungsprojekte → COKE

Systemsoftwareinfrastruktur heterogener Bildsysteme

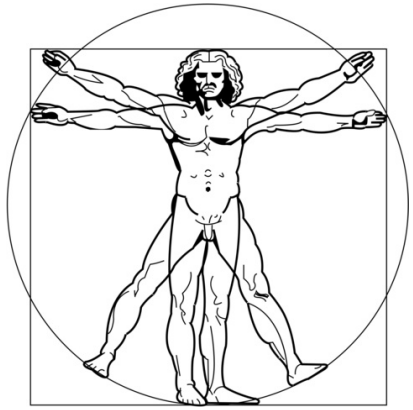
Resource-Aware Multi-Processing für heterogene Vielkernprozessoren



[4] `http://univis.uni-erlangen.de` → Forschungsprojekte → RAMP

Resource-Aware Multi-Processing für heterogene Vielkernprozessoren

- GPU-zentrische **Betriebsmittelverwaltung**
  - zeitlich berechenbares Laufzeitsystem
    - nichtverdrängbarer Kern (*run to completion*)
  - Periodisierung und Isolierung von GPU-Aufträgen
    - Einplanung nach Ausführungskosten
  - Abstimmung (*tradeoff*) von Durchsatz und Antwortzeit

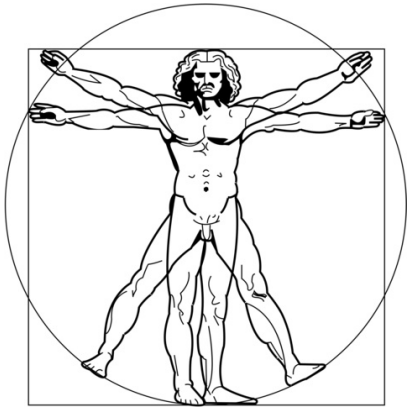


[4] `http://univis.uni-erlangen.de` → Forschungsprojekte → RAMP

Systemsoftwareinfrastruktur heterogener Bildsysteme

Resource-Aware Multi-Processing für heterogene Vielkernprozessoren

- GPU-zentrische **Betriebsmittelverwaltung**
  - zeitlich berechenbares Laufzeitsystem
    - nichtverdrängbarer Kern (*run to completion*)
  - Periodisierung und Isolierung von GPU-Aufträgen
    - Einplanung nach Ausführungskosten
  - Abstimmung (*tradeoff*) von Durchsatz und Antwortzeit



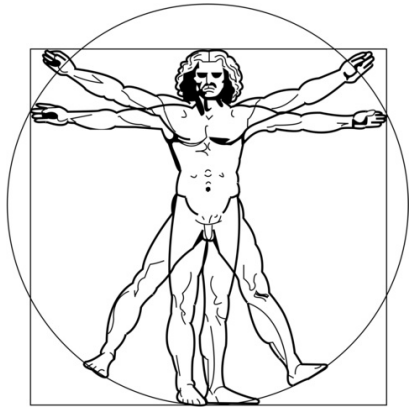
- RAM-zentrische **Laufzeitexekutive** für heterogene Vielkernsysteme
  - applikationsspezifische und datenstrukturorientierte Speicherverwaltung
  - Laufzeitanpassung und -relokation dynamischer Datenstrukturen

[4] `http://univis.uni-erlangen.de` → Forschungsprojekte → RAMP

Systemsoftwareinfrastruktur heterogener Bildsysteme

Resource-Aware Multi-Processing für heterogene Vielkernprozessoren

- GPU-zentrische **Betriebsmittelverwaltung**
  - zeitlich berechenbares Laufzeitsystem
    - nichtverdrängbarer Kern (*run to completion*)
  - Periodisierung und Isolierung von GPU-Aufträgen
    - Einplanung nach Ausführungskosten
  - Abstimmung (*tradeoff*) von Durchsatz und Antwortzeit



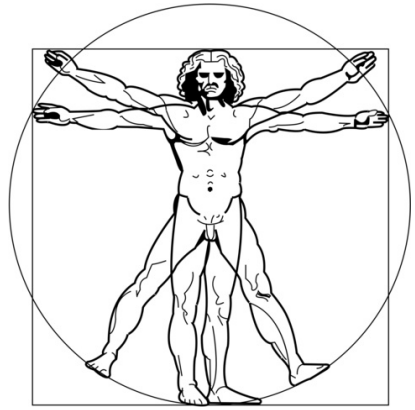
- RAM-zentrische **Laufzeitexekutive** für heterogene Vielkernsysteme
  - applikationsspezifische und datenstrukturorientierte Speicherverwaltung
  - Laufzeitanpassung und -relokation dynamischer Datenstrukturen
- für Bildsystemanwendungen **maßgeschneiderte Systemsoftware**
  - Unterstützung der inkrementellen Verbesserung visueller Qualität
  - Muster zur adaptiven Detailanpassung von Geometrie oder Texturen

[4] `http://univis.uni-erlangen.de` → Forschungsprojekte → RAMP

Systemsoftwareinfrastruktur heterogener Bildsysteme

Resource-Aware Multi-Processing für heterogene Vielkernprozessoren

- GPU-zentrische **Betriebsmittelverwaltung**
  - zeitlich berechenbares Laufzeitsystem
    - nichtverdrängbarer Kern (*run to completion*)
  - Periodisierung und Isolierung von GPU-Aufträgen
    - Einplanung nach Ausführungskosten
  - Abstimmung (*tradeoff*) von Durchsatz und Antwortzeit



- RAM-zentrische **Laufzeitexekutive** für heterogene Vielkernsysteme
  - applikationsspezifische und datenstrukturorientierte Speicherverwaltung
  - Laufzeitanpassung und -relokation dynamischer Datenstrukturen
- für Bildsystemanwendungen **maßgeschneiderte Systemsoftware**
  - Unterstützung der inkrementellen Verbesserung visueller Qualität
  - Muster zur adaptiven Detailanpassung von Geometrie oder Texturen

↪ 64-Kerner mit GPU-Erweiterung (Tesla Fermi, Kepler oder OptiX)

[4] `http://univis.uni-erlangen.de` → Forschungsprojekte → RAMP

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

[5] `http://univis.uni-erlangen.de` → Forschungsprojekte → AORTA

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

- **zeitgesteuerte** (*time-triggered*, TT) **Systeme**
  - Vorabwissen zwingend erforderlich
  - statische Ablaufplanung, vor Laufzeit
  - implizit koordinierte Prozesse, kein Laufzeitaufwand
  - von eher simpler Struktur, leichter analysierbar

[5] `http://univis.uni-erlangen.de` → Forschungsprojekte → AORTA

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

- **ereignisgesteuerte** (*event-triggered*, ET) **Systeme**
  - Vorabwissen nicht erforderlich, aber vorteilhaft
  - dynamische Ablaufplanung, zur Laufzeit
  - explizit zu koordiniernde Prozesse, Laufzeitaufwand
  - von eher komplexer Struktur, schwieriger analysierter

[5] `http://univis.uni-erlangen.de` → Forschungsprojekte → AORTA

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

- **zeitgesteuerte** (*time-triggered*, TT) **Systeme**
  - Vorabwissen zwingend erforderlich
  - statische Ablaufplanung, vor Laufzeit
  - implizit koordinierte Prozesse, kein Laufzeitaufwand
  - von eher simpler Struktur, leichter analysierbar
- **ereignisgesteuerte** (*event-triggered*, ET) **Systeme**
  - Vorabwissen nicht erforderlich, aber vorteilhaft
  - dynamische Ablaufplanung, zur Laufzeit
  - explizit zu koordiniernde Prozesse, Laufzeitaufwand
  - von eher komplexer Struktur, schwieriger analysierter

- **Migrationspfad** zwischen verschiedenen Echtzeitsystemarchitekturen
  - übersetzergestützte Transformation von ET- zu TT-Programmen *v.v.*
  - kanonische Programmierschnittstelle für Echtzeitanwendungssysteme

[5] `http://univis.uni-erlangen.de` → Forschungsprojekte → AORTA

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

- **zeitgesteuerte** (*time-triggered*, TT) **Systeme**
  - Vorabwissen zwingend erforderlich
  - statische Ablaufplanung, vor Laufzeit
  - implizit koordinierte Prozesse, kein Laufzeitaufwand
  - von eher simpler Struktur, leichter analysierbar
- **ereignisgesteuerte** (*event-triggered*, ET) **Systeme**
  - Vorabwissen nicht erforderlich, aber vorteilhaft
  - dynamische Ablaufplanung, zur Laufzeit
  - explizit zu koordiniernde Prozesse, Laufzeitaufwand
  - von eher komplexer Struktur, schwieriger analysierter

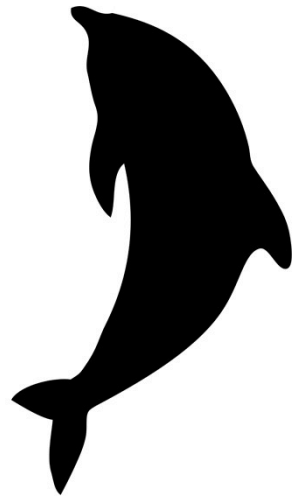
- **Migrationspfad** zwischen verschiedenen Echtzeitsystemarchitekturen
  - übersetzergestützte Transformation von ET- zu TT-Programmen *v.v.*
  - kanonische Programmierschnittstelle für Echtzeitanwendungssysteme

↪ verteilte Spezialzwecksysteme mehrkerniger Prozessoren

[5] `http://univis.uni-erlangen.de` → Forschungsprojekte → AORTA

Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

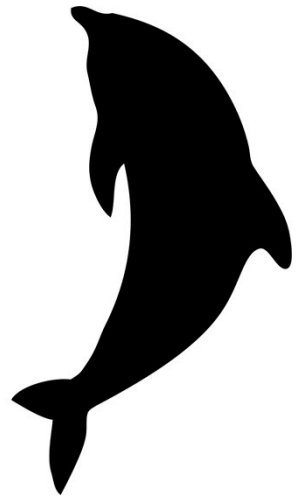


[6] `http://univis.uni-erlangen.de` → Forschungsprojekte → REFIT

Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

- **Virtualisierung** als Schlüsseltechnologie
  - Konsolidierung von Diensteeinheiten (*server*)
  - Redundanz durch replizierte virtuelle Maschinen

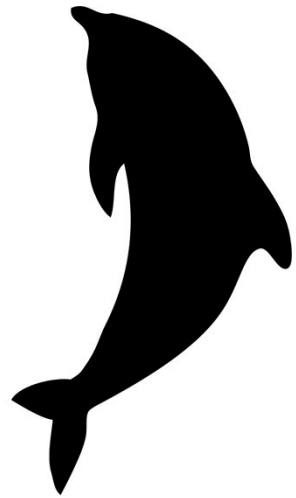


[6] `http://univis.uni-erlangen.de` → Forschungsprojekte → REFIT

Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

- **Virtualisierung** als Schlüsseltechnologie
  - Konsolidierung von Diensteeinheiten (*server*)
  - Redundanz durch replizierte virtuelle Maschinen
- **Vorhersage** wahrscheinlicher Ausführungspfade
  - deterministische mehrfädige Ausführung
    - geordnete Sperrreihenfolge (*lock sequence*)
    - Koordinierung zwischen Kopien als Ausnahmefall
  - skalierbare Sperrüberwachung und -verwaltung

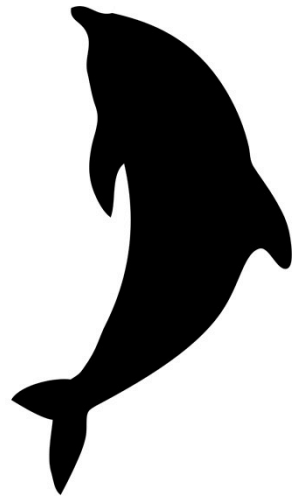


[6] `http://univis.uni-erlangen.de` → Forschungsprojekte → REFIT

Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

- **Virtualisierung** als Schlüsseltechnologie
  - Konsolidierung von Diensteeinheiten (*server*)
  - Redundanz durch replizierte virtuelle Maschinen
- **Vorhersage** wahrscheinlicher Ausführungspfade
  - deterministische mehrfädige Ausführung
    - geordnete Sperrreihenfolge (*lock sequence*)
    - Koordinierung zwischen Kopien als Ausnahmefall
  - skalierbare Sperrüberwachung und -verwaltung
- **minimal invasive Operation** für den Normalfall: $f + 1$ Kopien
  - dehnfähige (*resilient*) Einigungsprotokolle für den Ausnahmefall
  - performante Nachrichtenauthentifikation und -verifikation
  - zuverlässiger hardwarebasierter (FPGA) Zählerzuweisungsdienst

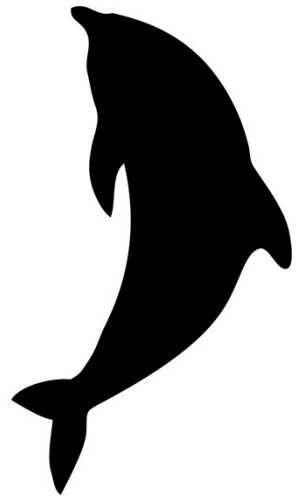


[6] `http://univis.uni-erlangen.de` → Forschungsprojekte → REFIT

Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

- **Virtualisierung** als Schlüsseltechnologie
  - Konsolidierung von Diensteeinheiten (*server*)
  - Redundanz durch replizierte virtuelle Maschinen
- **Vorhersage** wahrscheinlicher Ausführungspfade
  - deterministische mehrfädige Ausführung
    - geordnete Sperrreihenfolge (*lock sequence*)
    - Koordinierung zwischen Kopien als Ausnahmefall
  - skalierbare Sperrüberwachung und -verwaltung
- **minimal invasive Operation** für den Normalfall: $f + 1$ Kopien
  - dehnfähige (*resilient*) Einigungsprotokolle für den Ausnahmefall
  - performante Nachrichtenauthentifikation und -verifikation
  - zuverlässiger hardwarebasierter (FPGA) Zählerzuweisungsdienst



↪ massiv verteilte, mehrstufige (*multi-tier*) Systemarchitekturen

[6] `http://univis.uni-erlangen.de` → Forschungsprojekte → REFIT

Fäden als erstrangige Abstraktionen der Hardware

„Entabstrahierung“ funktioneller Merkmale des (realen) Prozessors

[7] `http://univis.uni-erlangen.de` → Forschungsprojekte → Sloth

Fäden als erstrangige Abstraktionen der Hardware

„Entabstrahierung“ funktioneller Merkmale des (realen) Prozessors

- **Latenzverbergung**
  - Unterbrecherhardware übernimmt die Einplanung und Einlastung von Fäden
  - Unterbrechungspriorität (IPL) und Fadenpriorität sind gleichgestellt, bilden denselben Prioritätenraum
  - eine Unterbrechungsanforderung (IRQ) kommt der Bereitstellung eines Fadens gleich

[7] `http://univis.uni-erlangen.de` → Forschungsprojekte → Sloth

Fäden als erstrangige Abstraktionen der Hardware

„Entabstrahierung“ funktioneller Merkmale des (realen) Prozessors

- **Latenzverbergung**
  - Unterbrecherhardware übernimmt die Einplanung und Einlastung von Fäden
  - Unterbrechungspriorität (IPL) und Fadenpriorität sind gleichgestellt, bilden denselben Prioritätenraum
  - eine Unterbrechungsanforderung (IRQ) kommt der Bereitstellung eines Fadens gleich
- **Prozessoranforderungen**
  - $number(IPL) \geq number(Threads)$
  - IRQ insbesondere auch durch Software auslösbar

↪ Infineon TriCore, ARM Cortex-M3

[7] `http://univis.uni-erlangen.de` → Forschungsprojekte → Sloth

Fäden als erstrangige Abstraktionen der Hardware

„Entabstrahierung“ funktioneller Merkmale des (realen) Prozessors

- **Latenzverbergung**
  - Unterbrecherhardware übernimmt die Einplanung und Einlastung von Fäden
  - Unterbrechungspriorität (IPL) und Fadenpriorität sind gleichgestellt, bilden denselben Prioritätenraum
  - eine Unterbrechungsanforderung (IRQ) kommt der Bereitstellung eines Fadens gleich
- **Prozessoranforderungen**
  - $number(IPL) \geq number(Threads)$
  - IRQ insbesondere auch durch Software auslösbar

  ↪ Infineon TriCore, ARM Cortex-M3
- werkzeuggestützter, **generischer Ansatz** der Systemprogrammierung
  - vollständige statische Konfigurierung, *vor* Lade- und Laufzeit

[7] `http://univis.uni-erlangen.de` → Forschungsprojekte → Sloth

Fäden als erstrangige Abstraktionen der Hardware

„Entabstrahierung“ funktioneller Merkmale des (realen) Prozessors

- **Latenzverbergung**
  - Unterbrecherhardware übernimmt die Einplanung und Einlastung von Fäden
  - Unterbrechungspriorität (IPL) und Fadenpriorität sind gleichgestellt, bilden denselben Prioritätenraum
  - eine Unterbrechungsanforderung (IRQ) kommt der Bereitstellung eines Fadens gleich
- **Prozessoranforderungen**
  - $number(IPL) \geq number(Threads)$
  - IRQ insbesondere auch durch Software auslösbar

  ↪ Infineon TriCore, ARM Cortex-M3

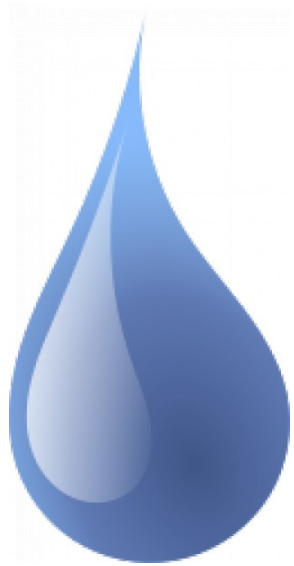
- werkzeuggestützter, **generischer Ansatz** der Systemprogrammierung
  - vollständige statische Konfigurierung, *vor* Lade- und Laufzeit

↪ homo-/heterogene mehrkernige Mikrosteuerrechner (*microcontroller*)

[7] `http://univis.uni-erlangen.de` → Forschungsprojekte → Sloth

Proaktive energiegewahre Systemprogrammierung

Gerüst (*framework*) zur Entwicklung energieeffizienter Programme



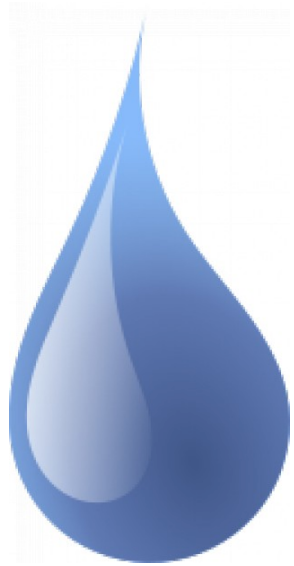
[8] `http://univis.uni-erlangen.de` → Forschungsprojekte → SEEP

Proaktive energiegewahre Systemprogrammierung

Gerüst (*framework*) zur Entwicklung energieeffizienter Programme

- **symbolische Programmausführung**
  - Erkundung (*exploration*) von Programmpfaden
  - Extraktion von zugehörigen Pfadbedingungen
  - Erzeugung funktional äquivalenter Binärprogramme

↪ nichtfunktionale Unterschiede bzgl. erw. Energiebedarf



[8] `http://univis.uni-erlangen.de` → Forschungsprojekte → SEEP

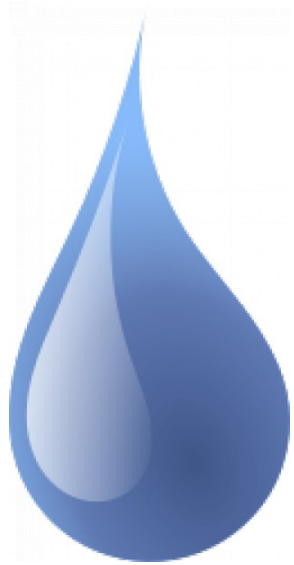
Proaktive energiegewahre Systemprogrammierung

Gerüst (*framework*) zur Entwicklung energieeffizienter Programme

- **symbolische Programmausführung**
  - Erkundung (*exploration*) von Programmpfaden
  - Extraktion von zugehörigen Pfadbedingungen
  - Erzeugung funktional äquivalenter Binärprogramme

  ↪ nichtfunktionale Unterschiede bzgl. erw. Energiebedarf
- **plattformspezifische Energieprofile**
  - spezifiziert den Energieverbrauch pro Maschinenbefehl
  - Verbrauchsbestimmung erfolgt effektiv pro Basisblock

  ↪ Erweiterung um E/A-Geräte bzw. -Schnittstellen



[8] `http://univis.uni-erlangen.de` → Forschungsprojekte → SEEP

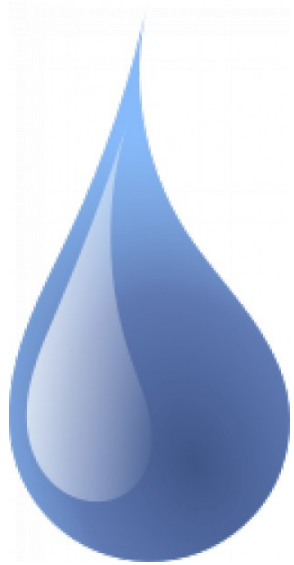
Proaktive energiegewahre Systemprogrammierung

Gerüst (*framework*) zur Entwicklung energieeffizienter Programme

- **symbolische Programmausführung**
  - Erkundung (*exploration*) von Programmpfaden
  - Extraktion von zugehörigen Pfadbedingungen
  - Erzeugung funktional äquivalenter Binärprogramme

  ↪ nichtfunktionale Unterschiede bzgl. erw. Energiebedarf
- **plattformspezifische Energieprofile**
  - spezifiziert den Energieverbrauch pro Maschinenbefehl
  - Verbrauchsbestimmung erfolgt effektiv pro Basisblock

  ↪ Erweiterung um E/A-Geräte bzw. -Schnittstellen
- analoge **Energiemessung** zur Validierung auf Basis eigener Hardware
  - elektrisches System frei von fehleranfälligen Abtastintervallen



[8] `http://univis.uni-erlangen.de` → Forschungsprojekte → SEEP

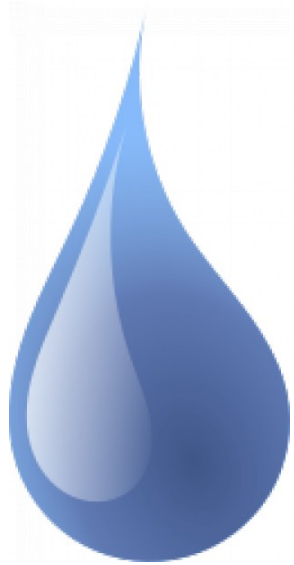
Proaktive energiegewahre Systemprogrammierung

Gerüst (*framework*) zur Entwicklung energieeffizienter Programme

- **symbolische Programmausführung**
  - Erkundung (*exploration*) von Programmpfaden
  - Extraktion von zugehörigen Pfadbedingungen
  - Erzeugung funktional äquivalenter Binärprogramme

  ↪ nichtfunktionale Unterschiede bzgl. erw. Energiebedarf
- **plattformspezifische Energieprofile**
  - spezifiziert den Energieverbrauch pro Maschinenbefehl
  - Verbrauchsbestimmung erfolgt effektiv pro Basisblock

  ↪ Erweiterung um E/A-Geräte bzw. -Schnittstellen
- analoge **Energiemessung** zur Validierung auf Basis eigener Hardware
  - elektrisches System frei von fehleranfälligen Abtastintervallen

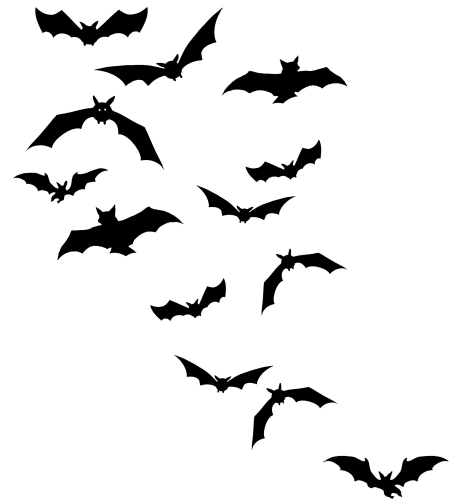


↪ homo-/heterogene vielkernige Prozessoren und HPC in Erwägung

[8] `http://univis.uni-erlangen.de` → Forschungsprojekte → SEEP

Softwareinfrastruktur vernetzter Sensorsysteme

Adaptive Run-Time Environment for Resource-scarce Sensor Systems

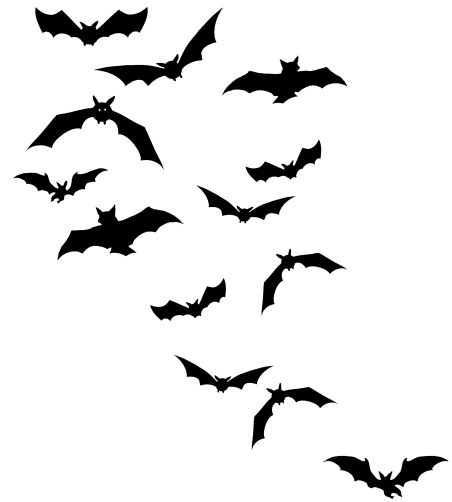


[9] `http://univis.uni-erlangen.de` → Forschungsprojekte → BATS

Softwareinfrastruktur vernetzter Sensorsysteme

Adaptive Run-Time Environment for Resource-scarce Sensor Systems

- **mobiles** (agiles) **Sensornetz** einerseits
  - Mausohr *Myotis myotis* mit „Sensorknotenrucksack“
    - Ortswechsel, Migration und Soziobiologie
  - Fliegengewicht (20 g), fordert extrem leichten Aufbau
    - Kleinstknoten (2 g) $\rightsquigarrow$ wenig Hardware, hochintegriert
  - Systemsoftware für winzigste Rechensysteme
    - extrem wenig Speicher und Rechenleistung
    - dynamisches Laden bei *ad-hoc* Netzwerkverbindungen
    - Betriebsdauer maximieren, Energieverbrauch minimieren

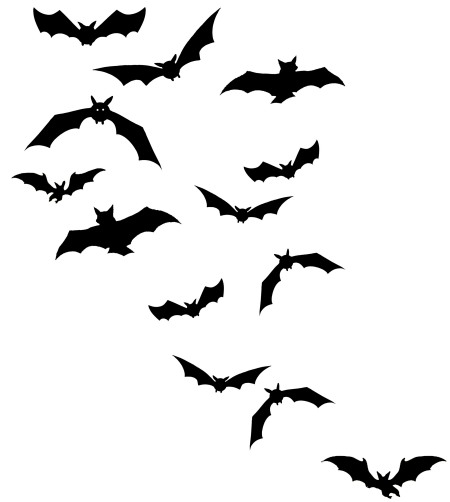


[9] `http://univis.uni-erlangen.de` → Forschungsprojekte → BATS

Softwareinfrastruktur vernetzter Sensorsysteme

Adaptive Run-Time Environment for Resource-scarce Sensor Systems

- **mobiles** (agiles) **Sensornetz** einerseits
  - Mausohr *Myotis myotis* mit „Sensorknotenrucksack“
    - Ortswechsel, Migration und Soziobiologie
  - Fliegengewicht (20 g), fordert extrem leichten Aufbau
    - Kleinstknoten (2 g) $\rightsquigarrow$ wenig Hardware, hochintegriert
  - Systemsoftware für winzigste Rechensysteme
    - extrem wenig Speicher und Rechenleistung
    - dynamisches Laden bei *ad-hoc* Netzwerkverbindungen
    - Betriebsdauer maximieren, Energieverbrauch minimieren



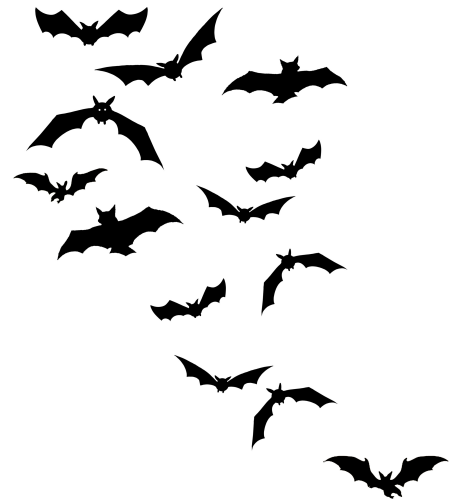
- **stationäres Sensornetz** andererseits: Netzkoppler (*gateway*)
  - spontane und kurzzeitige Bewirtung der vorbeifliegenden Sensorknoten

[9] `http://univis.uni-erlangen.de` → Forschungsprojekte → BATS

Softwareinfrastruktur vernetzter Sensorsysteme

Adaptive Run-Time Environment for Resource-scarce Sensor Systems

- **mobiles** (agiles) **Sensornetz** einerseits
  - Mausohr *Myotis myotis* mit „Sensorknotenrucksack“
    - Ortswechsel, Migration und Soziobiologie
  - Fliegengewicht (20 g), fordert extrem leichten Aufbau
    - Kleinstknoten (2 g) $\rightsquigarrow$ wenig Hardware, hochintegriert
  - Systemsoftware für winzigste Rechensysteme
    - extrem wenig Speicher und Rechenleistung
    - dynamisches Laden bei *ad-hoc* Netzwerkverbindungen
    - Betriebsdauer maximieren, Energieverbrauch minimieren



- **stationäres Sensornetz** andererseits: Netzkoppler (*gateway*)
  - spontane und kurzzeitige Bewirtung der vorbeifliegenden Sensorknoten

$\hookrightarrow$ „*Seeped Sloth*“ als minimale Basis von Systemfunktionen

$\hookrightarrow$ „*Multics-alike*“ dynamisches Binden als minimale Systemerweiterung

[9] `http://univis.uni-erlangen.de` $\rightarrow$ Forschungsprojekte $\rightarrow$ BATS

Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

[10] http://univis.uni-erlangen.de → Forschungsprojekte → ARES

Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

- **kombinierter Redundanzansatz**
  - holistische Absicherung v. Regelungsanwendungen
    - bestehend aus Sensorik, Regelung und Aktuatorik
    - *der* sicherheitskritische Teil des Gesamtsystems
  - softwarebasierte Fehlertoleranz
    - transiente Hardwarefehler tolerieren

[10] `http://univis.uni-erlangen.de` → Forschungsprojekte → ARES

Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

- **kombinierter Redundanzansatz**
  - holistische Absicherung v. Regelungsanwendungen
    - bestehend aus Sensorik, Regelung und Aktuatorik
    - *der* sicherheitskritische Teil des Gesamtsystems
  - softwarebasierte Fehlertoleranz
    - transiente Hardwarefehler tolerieren

- auf **gemischte Kritikalität** (*mixed criticality*) ausgerichtet
  - Echtzeitfähigkeit (weich, fest, hart) eng gekoppelt mit
  - Betriebs- (*safety*) und Angriffssicherheit (*security*)

[10] `http://univis.uni-erlangen.de` → Forschungsprojekte → ARES

Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

- **kombinierter Redundanzansatz**
  - holistische Absicherung v. Regelungsanwendungen
    - bestehend aus Sensorik, Regelung und Aktuatorik
    - *der* sicherheitskritische Teil des Gesamtsystems
  - softwarebasierte Fehlertoleranz
    - transiente Hardwarefehler tolerieren

- auf **gemischte Kritikalität** (*mixed criticality*) ausgerichtet
  - Echtzeitfähigkeit (weich, fest, hart) eng gekoppelt mit
  - Betriebs- (*safety*) und Angriffssicherheit (*security*)
- **Messunsicherheit** refl. Entwurfskonzept für Regelungssysteme
  - problemspezifische Modularisierung von Regelungsanwendungen
  - Aufbau zuverlässiger System aus unzuverlässigen Hardwarekomponenten

[10] `http://univis.uni-erlangen.de` → Forschungsprojekte → ARES

Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

- **kombinierter Redundanzansatz**
  - holistische Absicherung v. Regelungsanwendungen
    - bestehend aus Sensorik, Regelung und Aktuatorik
    - *der* sicherheitskritische Teil des Gesamtsystems
  - softwarebasierte Fehlertoleranz
    - transiente Hardwarefehler tolerieren

- auf **gemischte Kritikalität** (*mixed criticality*) ausgerichtet
  - Echtzeitfähigkeit (weich, fest, hart) eng gekoppelt mit
  - Betriebs- (*safety*) und Angriffssicherheit (*security*)
- **Messunsicherheit** refl. Entwurfskonzept für Regelungssysteme
  - problemspezifische Modularisierung von Regelungsanwendungen
  - Aufbau zuverlässiger System aus unzuverlässigen Hardwarekomponenten

↪ Quadrocopter (I4Copter), CiAO

[10] `http://univis.uni-erlangen.de` → Forschungsprojekte → ARES

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

- ressource-effiziente, **spekulative Fehlertoleranz**
  - leichtgewichtige, transaktionale Kontrollflussaspekte
  - im BS umgesetzt als Fortsetzung (*continuation*)

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

- ressource-effiziente, **spekulative Fehlertoleranz**
  - leichtgewichtige, transaktionale Kontrollflussaspekte
  - im BS umgesetzt als Fortsetzung (*continuation*)
- aspektorientierte Programmierung (**AOP**)
  - *n*-modulare Redundanz (NMR) als Erweiterungsaspekt
  - Absicherung dynamischer Bindung (OO, *vptr guard*)

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

- ressource-effiziente, **spekulative Fehlertoleranz**
  - leichtgewichtige, transaktionale Kontrollflussaspekte
  - im BS umgesetzt als Fortsetzung (*continuation*)
- aspektorientierte Programmierung (**AOP**)
  - *n*-modulare Redundanz (NMR) als Erweiterungsaspekt
  - Absicherung dynamischer Bindung (OO, *vptr guard*)
- (semi-) **automatische Programmanalyse**
  - zur Bewertung der Schadanfälligkeit von Software
  - statische Analyse, ergänzt um dynamische Analyse

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

- ressource-effiziente, **spekulative Fehlertoleranz**
  - leichtgewichtige, transaktionale Kontrollflussaspekte
  - im BS umgesetzt als Fortsetzung (*continuation*)
- aspektorientierte Programmierung (**AOP**)
  - *n*-modulare Redundanz (NMR) als Erweiterungsaspekt
  - Absicherung dynamischer Bindung (OO, *vptr guard*)
- (semi-) **automatische Programmanalyse**
  - zur Bewertung der Schadanfälligkeit von Software
  - statische Analyse, ergänzt um dynamische Analyse
- generischer NMR-Ansatz durch **typsichere Programmiersprache**
  - kompilierergestützte Instanzenbildung der erforderlichen Kopien (*replica*)
  - automatische Regenerierung abgeschlossener Zustandshüllen (*closure*)

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Aspektorientierte Systemsoftwarezuverlässigkeit

Dependability Aspects in Configurable Embedded Operating Systems

- ressource-effiziente, **spekulative Fehlertoleranz**
  - leichtgewichtige, transaktionale Kontrollflussaspekte
  - im BS umgesetzt als Fortsetzung (*continuation*)
- aspektorientierte Programmierung (**AOP**)
  - *n*-modulare Redundanz (NMR) als Erweiterungsaspekt
  - Absicherung dynamischer Bindung (OO, *vptr guard*)
- (semi-) **automatische Programmanalyse**
  - zur Bewertung der Schadanfälligkeit von Software
  - statische Analyse, ergänzt um dynamische Analyse
- generischer NMR-Ansatz durch **typsichere Programmiersprache**
  - kompilierergestützte Instanzenbildung der erforderlichen Kopien (*replica*)
  - automatische Regenerierung abgeschlossener Zustandshüllen (*closure*)

↪ Quadrocopter (I4Copter), KESO, CiAO

[11] `http://univis.uni-erlangen.de` → Forschungsprojekte → DanceOS

Literaturverzeichnis I

[1] Lohmann, D. ; Schröder-Preikschat, W. :
Betriebssysteme.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_BS, 2008 *ff.*

[2] Schröder-Preikschat, W. ; Kleinöder, J. :
Systemprogrammierung.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_SP, 2008 *ff.*