Real-Time Computing and the Evolution of Embedded System Designs

Tei-Wei Kuo (Award Recipient), Jian-Jia Chen, Yuan-Hao Chang, and Pi-Cheng Hsiu
Thank you the Real-Time Systems Community for treating me like a family member!

Paris, 2017
Real-Time Computing and Embedded Systems

The field of **real-time computing** is rich in research problems!
- More specific in their applications
- More drastic for their failures

An **embedded system** is a programmed controlling and operating system with a dedicated function within a larger mechanical or electrical system, often with **real-time computing constraints**. (Wikipedia)
Real-Time Computing

- System Correctness:
  - Logical Correctness ("the results are correct")
  - Temporal Correctness ("the results are delivered in/on time")

- High reactivity and high dependability are more important than the average performance

- Many Results in Real-Time Computing:
  - Least Upper Bound of Utilization Factor
  - Synchronization and Priority Ceiling
  - More Flexible Task Models, e.g., Multi-Frame Tasks
Timing correctness is the key factor to justify whether the system is safe or not. For hard real-time systems, since “any deadline miss can jeopardize the entire system,” it is not allowed to have any deadline miss.

REAL-TIME COMPUTING
Execution Time Depends upon

- The input, determining which path is taken
- The state of the hardware platform:
  - Due to caches, pipelining, speculation, etc.
- Interference from the environment:
  - External interference as seen from the analyzed task on shared buses, caches, memory

Figures from Jan Reineke and Reinhard Wilhem
Worst-Case Execution Time (WCET)

- Fundamental Research in Real-Time Systems
  - Active research topic ever since scheduling is explored!
  - Rich Literature in Uniprocessor Systems
  - Commercial Tools, Industrial Case Studies, etc.

- Significant Influence over Multicore Systems:
  - Popular Topic Regularly Being Seen as Sessions in Real-Time Conferences
  - Significant Impacts on the Advance in Using Multicore Platforms in Real-Time Computing

- Radojkovic et al. (ACM TACO, 2012) on Intel Atom and Intel Core 2 Quad: Up to 14x Slow-Down, Due to Interference on Shared L2 Cache and Memory Controller
Energy-Efficiency versus Exec Time

Dynamic power consumption at speed/frequency $s$ GHz
1.52s³ Watt

Static power consumption
0.08 Watt

execute at 0.297 GHz for 3.37 seconds
minimize the overall energy

arrival time

deadline

sleep

3.37sec

time

Energy minimization while satisfying the real-time constraints
- Active research topics since 2000
- Thermal behavior analysis under the real-time constraints since 2005
- In both cases, time is the major constraint
How about Soft Real-Time Computing

Rare deadline misses are often acceptable!

Industrial safety standards ~ failures under certain probability
- IEC-61508: Safety Standard for Electronics
- ISO-26262: Safety Standard for Automotive Systems

Safe Upper Bound

Mixed of Hard and Soft Real-Time Tasks: Reservation!

Guaranteed Isolations for Hard Real-Time Tasks
Proved Progressiveness for Soft Real-Time Tasks

Fixed-Priority Servers: Polling Server, Periodic Server, Sporadic Server, Deferrable Server, etc.

Dynamic-Priority Servers: Total bandwidth server (TBS), Constant bandwidth server (CBS), Proportional Share (PS), etc.
In contrast to real-time computing with time as the key factor, “time” becomes a *feature* in embedded system designs.
Computing with Human

<table>
<thead>
<tr>
<th>Human Perception</th>
<th>User Perception over Display, Sound, and More</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interactivities</td>
<td>User-Centric Resource Support over Embedded Systems</td>
</tr>
<tr>
<td>User Attention</td>
<td>Perceived and Unperceived Activities over Embedded Systems</td>
</tr>
</tbody>
</table>
Paradigm Shift in Computing

- User Behavior (Diversity)
- Application Semantics (Variety)
- Device Features (Distinctivity)
User-Centric Task Scheduling

Performance Metrics

- Energy Efficiency
- User Experience (a variant of “time”)

Needs to Resource Reservation

- Require ways to reserve computing resource to applications in a way “proportionally” to user attention
- Applications must be executed and scheduled to improve energy efficiency and user experience

Content-Aware Resource Allocation

- Increasingly high resolution and frame rate
  - Not always with improved perceptual quality
- Complementary energy savings over DVFS by reducing the GPU workloads
  - Dynamic resolution scaling (w.r.t viewing distance or scrolling speed)
  - Dynamic frame rate scaling (redundant frames)
- Content-aware resource allocation
  - The time required to render a frame depends on the quality of contents perceived by the user
  - The deadline in rendering a frame depends on the frame rate required by the user
  - How to schedule tasks with dynamically adjustable execution times and deadlines?

Dynamic Resolution Scaling

Dynamic frame rate scaling
Attention-Based Resource Allocation

Background activities **imperceptibly** drains batteries

- **Repeating Interval**: static (periodic) or dynamic (sporadic)
- **Execution Windows**: within which to execute an activity (deadline)

**Activity alignment**

- **Example**: A1 (*perceptible* HW) and A3 (*imperceptible* HW) have overlapped execution windows, while A2 and A3 require the same *imperceptible* HW

**Observation**: HW similarity reflects the degree of energy savings, while time similarity reflects the impact on user perception

Native activity alignment

- **Activity 1**: 
  - **Vibrator**
  - **400 mJ**

- **Activity Manager**
  - **Queue**

- **Activity 2**: 
  - **WPS**
  - **3,650 mJ**

- **Activity 3**: 
  - **WPS**
  - **3,650 mJ**

Total energy: **7,520 mJ**

Similarity-based activity alignment

- **Activity 1**: 
  - **Vibrator**
  - **400 mJ**

- **Activity Manager**
  - **Queue**

- **Activity 2**: 
  - **WPS**
  - **3,650 mJ**

- **Activity 3**: 
  - **WPS**
  - **3,650 mJ**

Total energy: **4,050 mJ**

Huge Driving Forces

BMI Data

More than Moore

Tei-Wei Kuo, NTU
Challenges in Computing

Cloud Storage

Storage

Main Memory

CPU

GPU

AI Chips

Intelligence

Analysis

New
Ways to Break Memory Boundaries

- **Performance**
  - The gaps of memories is closer than ever.

- **Capacity**
  - They all grow at paces faster than Moore’s Law.

- **HW/SW**
  - Boundaries are blurring or shifting.
Innovation to Reshape Storage and Computing Markets

- Tremendous Performance Gap between the Main Memory and Storage
- Huge Barrier to Move Data from the Memory to Computing Units

Unified Memory

... Persistent ...

Huge Addressing Space

Process-in-Memory
Between Main Memory and Storage

Big Data to Cross the Tremendous Gap between the Main Memory and Storage

Applications

User Space
- Process 1
- Process 2
- ...
- Process n

Operating System

Kernel Space
- Virtual Memory
- File System
- Device Driver

Software

Hardware

Integrated Memory Devices (Pure PCM)

Caching Again: WCET Issue Only?

- Another Dimension in Designs
  - Endurance
  - Read/write asymmetry of NVM

Existing caching algorithms considers performance. The caching algorithms for NVM-based systems need to consider read/write asymmetry and endurance issues.

Huge Barrier to Move Data from the Memory to Computing Units

- Scalability of Existing AI Solutions?
- Machine learning requires high memory bandwidth

![Graph showing memory bandwidth requirement over time](image)

Deng et al., “Reduced-Precision Memory Value Approximation for Deep Learning”, HPL Report, 2015

High Bandwidth Memory: The Great Awakening of AI, 2018
Huge Barrier to Move Data from the Memory to Computing Units

- Process-in-Memory (PIM) to resolve the memory bandwidth issue.

- Analog variation error caused by programming variation of crossbar memories

Huge Barrier to Move Data from the Memory to Computing Units

- Design issues of data placement and data flow with input/output buffers in PIM.
- Algorithm modification for workload partition between CPU/GPU and crossbar PIM memory.
- Algorithm modification to fit in the special characteristic of PIM.

Systolic architecture
The advances in mobile systems, memory innovations, and use cases have inspired the evolution of embedded system designs and insights to solutions regarding how systems should be restructured and how computing should be done.

RETHINKING REAL-TIME COMPUTING WITH EMBEDDED SYSTEM EVOLUTION
The Internet-of-Thing Era

- Unstable Energy Sources
- Normally-Off Computing

Intermittent Computing!

RF: Unstable  Thermal: Relatively stable  Solar: Environment dependent

100x decrease in volume every decade

Program Progress (%)
0 50
ON OFF
Intermittent execution

Failure  Resume
Roll back

Volatile processor
Progress
Emerging of Non-Volatile Computing/Memory Devices

- Performance Metrics: maxspan vs. forward progress
- Schedulability tests with power failure possibility

Data Integrity

Battery-less wearable
Battery-less mobile phone
Boundary Breaking between Computing Units and Memory

- Advances in manufacturing and devices presents huge performance gaps between traditional system layers.
- Do we need new task models in computing and scheduling/analysis methodologies?
Although many successful stories can be told to design embedded systems with technology developed in real-time systems, some limitation of our research efforts in real-time systems is foreseen and must be further exploited in designing advance embedded systems.

OUR PERSPECTIVES
Successful Stories and Limitation

Many Successful Results and Applications

- Fixed-priority schedulers in almost every RTOS
- EDF in some RTOSes
- PIP and PCP as part of POSIX
- The application of real-time technology in control area network (CAN)
- WCET analyzer adopted in the industry

However…

- Computing systems are getting more and more complex
- Designing only for the worst case might become a design bottleneck and only applicable for highly reliable systems.
- The industry seems adopting only a small portion of our work
- ....
Then...

Huge tsunami of computer system revolution is coming!
謝謝！Xièxie!
Thank You!

國立臺灣大學
National Taiwan University