Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 1891 – 1898

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

Scheduling Challenges in Mixed Critical Real-time Heterogeneous Computing Platforms

Chetan Kumar N G[a], Sudhanshu Vyas[a], Ron K. Cytron[b], Christopher D. Gill[b], Joseph Zambreno[a], Phillip H. Jones[a,1]

[a]*Iowa State University, Department of Electrical and Computer Engineering, Ames, IA*
[b]*Washington University, Computer Science and Engineering, St. Louis, MO*

Abstract

In Dynamic Data-Driven Application Systems (DDDAS), applications must dynamically adapt their behavior in response to objectives and conditions that change while deployed. Often these applications may be safety critical or tightly resource constrained, with a need for graceful degradation when introduced to unexpected conditions. This paper begins by motivating and providing a vision for a dynamically adaptable mixed critical computing platform to support DDDAS applications. We then specifically focus on the need for advancements in task models and scheduling algorithms to manage the resources of such a platform. We discuss the short comings of existing task models for capturing important attributes of our envisioned computing platform, and identify challenges that must be addressed when developing scheduling algorithms that act upon our proposed extended task model.

Keywords: mixed criticality, real-time system, hybrid computing, hardware accelerators

1. Introduction

An applications ability to respond dynamically and swiftly to new information is central to the Dynamic Data Driven Applications Systems (DDDAS) concept. To achieve this capability, run-time platforms are needed that can monitor and adapt to changing conditions, not only with respect to an applications evolving requirements, but also to the dynamics of the platform and the operating environment.

While conventional techniques have optimized such platforms for performance, a greater challenge today is to optimize the platforms ability to *anticipate strategic surprise* [1]. Success is then measured by the platforms ability to retask themselves in response to unexpected phenomena that spontaneously introduce requirements to monitor, avoid, or respond to such surprises. This challenge falls squarely in the domain of DDDAS and is especially important for assets that are in flight and therefore typically beyond the reach of physical modification.

Towards the development of such systems, our group's research aims to develop a prototype execution framework through which application and environment data are streamed at run-time, and which can adapt dynamically

Email addresses: cknng@iastate.edu (Chetan Kumar N G), spvyas@iastate.edu (Sudhanshu Vyas), cytron@cse.wustl.edu (Ron K. Cytron), cdgill@cse.wustl.edu (Christopher D. Gill), zambreno@iastate.edu (Joseph Zambreno), phjones@iastate.edu (Phillip H. Jones)

[1]Corresponding author

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.358

to maintain essential system properties, and to optimize the collection, analysis, and application of the data. In addition to adapting its own behavior to the data flowing through it, the execution framework will shape and optimize the flows of data themselves, to integrate multiple concerns of the DDDAS platform and its applications.

Illustrative example. Consider an autonomous helicopter operating within specified mission parameters, such as the requirement to acquire and maintain surveillance of particular ground vehicles. The data streams that flow through the system may include: 1) data from sensors that capture the state of the environment in which the helicopter is operating, 2) data from sensors that capture the physical state of the helicopter, 3) monitors that assess and track the health and performance of power and computational electronics, and 4) monitors that track and quantify how well the helicopter is performing tasks specific to the mission at hand.

However, since the very nature of surprise precludes its precise characterization *a priori*, a significantly more comprehensive and fundamental shift in how the resources of the mission platform may be retasked to address surprise is needed. In response to unexpected adverse conditions, tactical opportunities, or in-mission re-prioritization of objectives, the sensors, computational resources, and flight-control systems may require different combinations of coordination with respect both to their individual behavioral requirements and to cross-cutting constraints on overall system properties. For example, the filters that select and transform data may require reprogramming and reconfiguration so that an unexpected phenomenon of interest can be monitored, computational resources may require reallocation to ensure timely extraction of mission-relevant information from that data, and flight control parameters may require adaptation to keep the helicopter oriented to best observe the phenomenon while it persists.

To illustrate how data streaming through a DDDAS execution framework could dynamically optimize performance in the helicopter example, consider how an on-board camera could be used to track an object entering its field of vision. To ensure both overall image quality and the accuracy of object identification and tracking in particular, it is necessary to ensure that the helicopter's flight control algorithm keeps the helicopter reasonably steady. Such a control function would be designed to meet a particular set of specifications (*e.g.,* hold the helicopter within k meters of position (x, y, z), with an angle of deviation from level of no more than θ).

Although this kind of specification and applicable control theory are both well understood, in practice there are a number of dynamic factors that can impact both the effectiveness and precision of such control. For example, coaxial electric helicopters may suffer blade strikes that can cause a part of the main rotor blade to chip. Even a small chip on a rotor blade in turn can have a significant impact on flight dynamics for which the vehicle's flight controllers, platform resources, and applications may need to compensate.

Furthermore, once it has been determined that the helicopter has a damaged blade, to maintain both control of the vehicle and awareness of its condition, the execution framework may need to modify how sensor data is streamed through the system. For example, when the helicopter is flying smoothly with undamaged blades, using a light-weight filter for the accelerometer sensor data may be sufficient (*e.g.,* computing a weighted running average). However, in the presence of heightened vibrations a more computationally expensive filter that fuses data from accelerometer and gyroscope sensors maybe needed (*e.g.,* a Kalman filter).

Limitations of the current state of the art. At issue in the example above is the extent to which such diverse modifications to a mission can be characterized and the relevant hardware and software reconfigured to meet its constraints and accomplish its goals reliably, under a wide range of such possible adaptations. Both the original and modified missions would typically contain elements that classically would be called embedded (small footprint), real-time (must meet deadlines and have low latency), and high-performance (best possible throughput, data fidelity, and computational resolution). While it may have been possible pre-flight to analyze the original mission with respect to the platform's ability to operate stably and meet its requirements, relatively little time may be available to determine the suitability of the platform for its modified mission once a phenomenon of interest appears.

While the DDDAS paradigm appears well suited for addressing a number of aspects of the previously given example, traditional system design tools and methods treat disjointly the behaviors of individual hardware and software components, and the many system properties that cross-cut them, as Figure 1 illustrates. Traditional layers of abstraction also tend to isolate application-level concerns from hardware- and physical-level concerns.

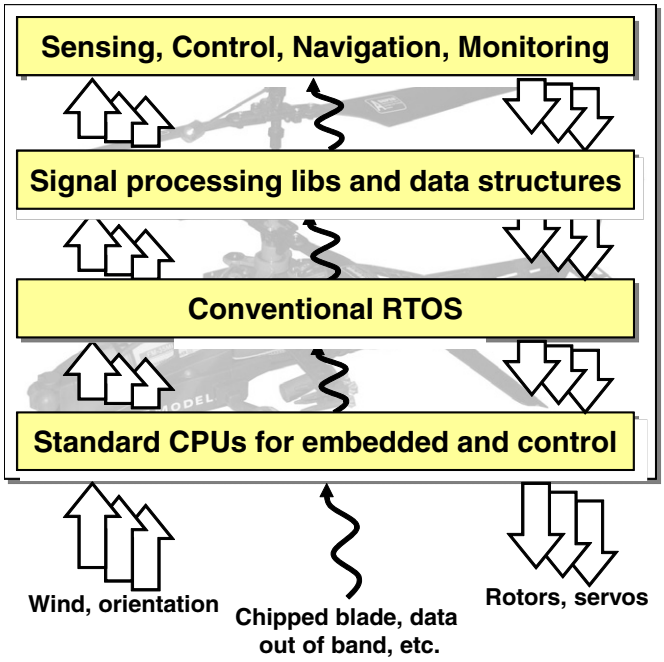

Fig. 1: Traditional layered architecture for helicopter example.

While this can help developers of traditional applications manage system complexity, such separation comes at a cost of less visibility and control over the interactions among the hardware and software components.

Grand Vision. The current state of the art in system software and hardware platforms poses crucial impediments to realizing the full potential of the DDDAS paradigm, and motivates an ambitious reconsideration of how hardware and software co-design can enable coordinated adaptive re-configuration while a mission is in progress.

The grand vision of our work is to move toward producing a computing platform that seamlessly conjoins system properties that are typically treated disjointedly (*e.g.,* real-time schedulability, and feedback from internal and external sensors). If successful, this holistic approach to cross-cutting system properties and the exploration of underlying enhancements to compilers, middleware, and computer architecture would reinforce and enhance DDDAS ability to optimize system performance through the interplay of streamed data and dynamic execution.

Figure 2 depicts the high-level vision we have for weaving system-level properties and concerns throughout the system stack. A significantly more dynamic treatment of system behavior and properties is enabled by our envisioned execution framework, which orchestrates the migration of functionality and sharing of information across computing layers, for the purpose of increased system efficiency and robustness. Middleware is tuned to application needs through dynamic data structure adaptation and a property-aware scheduler, while hardware resources are retasked under the direction of middleware to best serve system demands.

The system stack will support data structures whose properties are dynamically adapted in response to streamed data to make performance trade-offs at the system level (*e.g.,* timing jitter vs. memory footprint vs. throughput vs. thermal and power concerns). Continuing on this front, middleware and compiler mechanisms will support aspect-based weaving of system properties into these data structures. Additionally, low-level hardware-based system monitors and instruction set architectures will support low-latency dynamic adaptation to changes in streamed data from sensors and hardware monitors, thus integrating both hardware and software layers of an overall DDDAS architecture.

New task models and scheduling algorithms. This paper focuses on the need for advancements in task models and scheduling algorithms to support our envisioned platform's adaptation to *strategic surprise*. These models

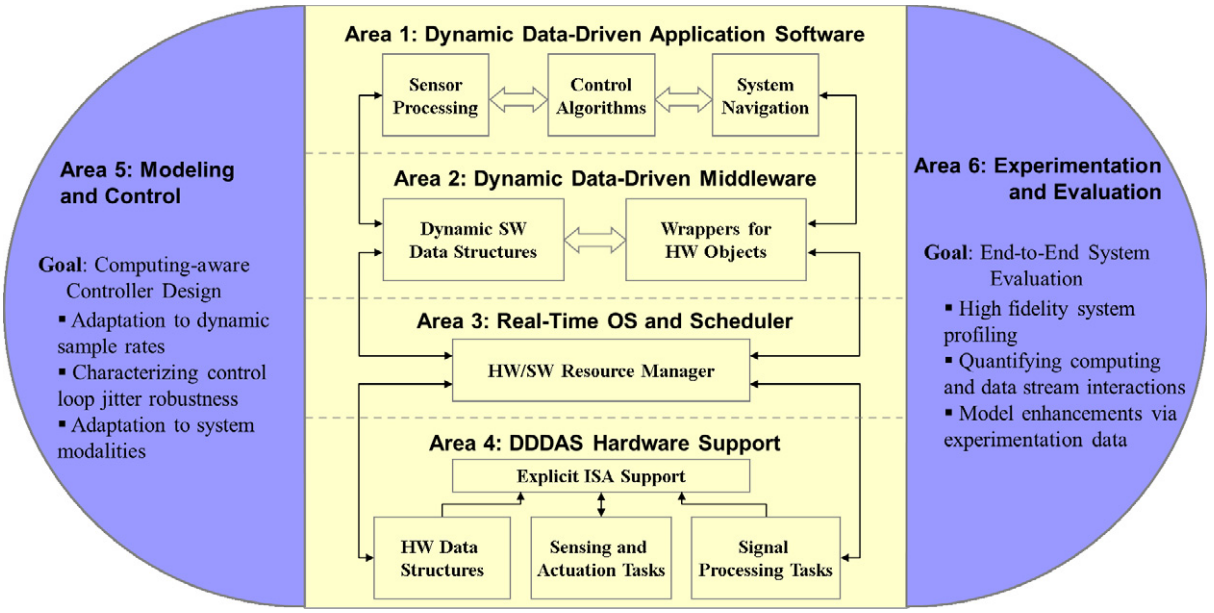

Fig. 2: Integrated hardware/software architecture for adaptive reconfigurable execution.

and algorithms will form the heart of our platform's "Schedule" and "HW/SW Resource Manager". They must capture the dynamics of mission mode changes, the existence of heterogeneous computing resources that are shared among many tasks, and allow for graceful degradation under overload conditions. The remainder of this paper discusses: 1) short comings of existing models for developing schedulers for such mixed critical real-time heterogeneous computing platforms, and 2) challenges that must be addressed by algorithms that are applied to our proposed extended task model.

2. Related Work

To address the issue of scheduling in mixed criticality systems, two alternative task models have been proposed by Vestal et al. [2, 3] and de Niz et al. [4, 5]. In Vestal's multi-criticality task model, each task τ_i is assigned a criticality level L_i and may have alternative worst case execution times (WCET), $C_i(l)$, corresponding to different criticality levels. The higher the criticality level, the more conservative will be the WCET estimation. Vestal et al. suggested the use of Audesly's prioirity assignment scheme [6] and period transformation technique [7] to improve the schedulability and utilization of mixed criticality tasks. Many scheduling models and algorithms [8, 9, 10, 11, 12] have been proposed based on Vestal's model to improve the schedulability of certifiable mixed criticality tasks. Effectiveness of reservation-based and priority-based scheduling approaches to dual-criticality systems were studied in [8, 11] by using processor speed-up factor as a metric. PLRS, a scheduling algorithm for certifiable mixed criticality sporadic task systems is presented in [9] and an offline computation method was provided to check the schedulability of the task set. Criticality based earliest deadline first (CBEDF) algorithm was presented in [10] to schedule tasks on dual-criticality systems. Earliest Deadline First with Virtual Deadlines (EDF-VD) scheduling algorithm was proposed in [12] for scheduling of mixed-criticality implicit-deadline sporadic tasks on preemptive uniprocessors.

In de Niz's task model, each task τ_i can have two execution times: C_i - worst case execution time under normal conditions and C_i^o - overload execution budget. Each task is assigned a criticality level L_i. Based on this task model, a zero slack scheduling method was proposed by de Niz et al. [4], which works on top of any priority based scheduling algorithm. Each task can be executed in either normal or critical mode. The tasks in normal mode are scheduled based on their priority to maximize resource utilization. When executing in critical mode, all the lower criticality tasks are suspended to guarantee the execution of higher criticality tasks. A metric

for overload-resilience called ductility was developed and Compress-On-Overload Packing (COP), an algorithm which works on top of zero slack rate monotonic scheduler to maximize ductility in distributed mixed-criticality systems was presented in [5].

The scheduling algorithms discussed above do not apply when tasks share mutually exclusive resources. Lakshmanan et al. [13] presented extensions to priority inheritance and ceiling protocols for zero slack scheduling [4] to solve the task synchronization problem in mixed criticality systems. A two tier dynamic resource management framework for mixed criticality embedded systems is presented in [14]. The framework provides static resource guarantees and enables fault isolation for distributed application subsystems with mixed criticality requirements and facilitates certification of safety-critical applications. A software based memory throttling mechanism is presented in [15], which controls the memory interference and guarantees the schedulability of critical tasks in mixed criticality real-time systems.

3. Mixed Critical Real-time Heterogeneous Computing Platforms: Proposed Task Model

Mixed critical real-time heterogeneous systems execute under varied operating conditions. Identifying different system configurations which is representative of all the scenarios and operating conditions would be challenging if not unrealistic. However, we believe it is possible to identify some of the key configurations or modes of operation during the design phase and transitions between these modes could be tested and certified. We call these as stable states of execution. As an example let us consider an autonomous helicopter with a mission to acquire and maintain surveillance of a ground vehicle. The stable states of execution and the occurrence of "surprise" during a representative autonomous helicopter mission is illustrated in Figure 3. The autonomous aircraft needs to minimize the consequences of these surprise situations. This could be achieved by dynamically changing the application characteristics to maintain the stability of the system. A task model should capture these dynamic changes in operating conditions to support the platform's ability to anticipate strategic surprise.

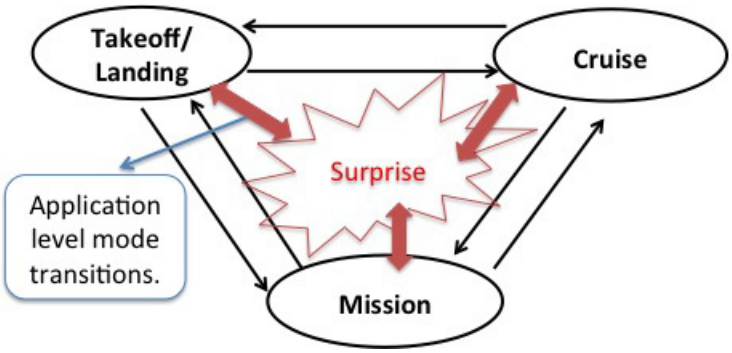

Fig. 3: Illustration of a "surprise" occurring during a representative autonomous helicopter mission.

The following are some of the main properties of mixed critical heterogeneous computing platforms which are not captured in the existing mixed criticality task models:

Dynamically changing task criticality. Existing mixed criticality models assume, the criticality level of a task to be constant. Some tasks may be critical only during certain operating conditions. Assuming the task to be critical all the time may lead to under utilization of resources as more conservative WCETs need to considered for schedulability analysis. A better utilization of resources could be achieved by varying the criticality of the tasks based on operating conditions.

Required and Optional Resources. Conventional resources, such as data structures and files, are considered as required resources. Some resources could be optional and its availability may increase application performance, predictability and/or improve quality of service. Reconfigurable hardware accelerators are an example of an optional resource. The availability of these optional resources enable the scheduler to optimize resource sharing for different system attributes such as stability, utilization and/or other utility functions.

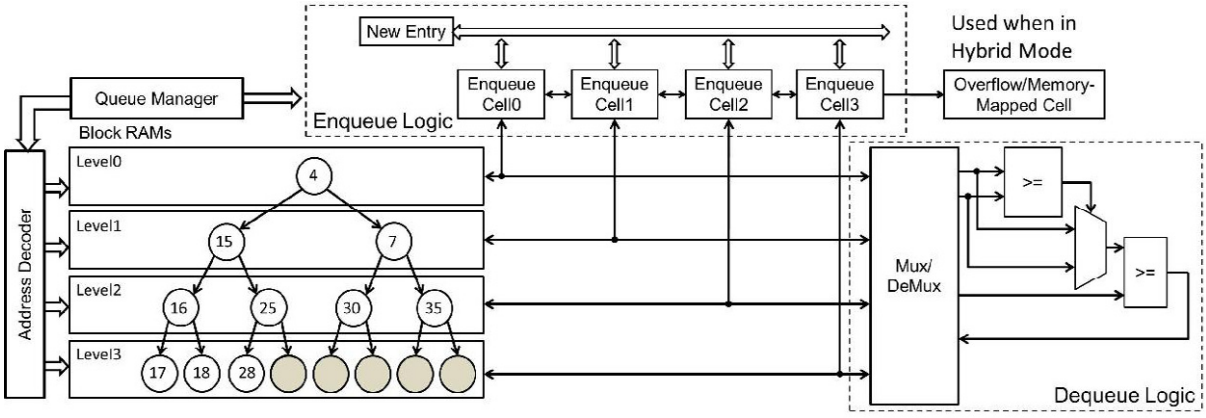

Fig. 4: Hardware priority queue architecture. As queue data is stored in hardware, allocating the hardware priority queue to a different task will incur considerable overhead. This should be accounted for during schedulability analysis.

Resource Preemption Overhead. Generally, overhead incurred due to preemption of resources is not accounted for (e.g. placing a semaphore on a data structure.). Some resources (e.g. hardware accelerators) may have large preemption overhead and this needs to be accounted for accurate schedulability analysis. For example, consider the hardware priority queue described in [16]. A high-level hardware architecture diagram of the priority queue is shown in Figure 4. For example, lets say that the hardware priority queue is being used by the scheduler. The queue data will be stored in hardware as shown in Figure 4. Now to allocate the queue to a another application (e.g. network bandwidth manager), the scheduler data stored in hardware needs to be copied to software memory and the hardware queue should be initialized with new data. The preemption overhead of the hardware priority queue varies depending on the size of the queue. This cannot be ignored during scheduling/resource allocation.

WCET dependency on resource allocated. The availability of reconfigurable logic will enable the use of hardware accelerators to improve application performance and predictability. The resources allocated and, in turn, a task's WCET can change during run time.

Proposed Task Model. Taking into consideration the properties discussed, a new task model is presented where each task, τ_i, is defined as:

$\tau_i = (T_i, A_i, D_i, P_i, V_i, RR_i[], L_i, M_i, C_i(RA_i[]))$ where,

- T_i is the task period,
- A_i is the arrival time,
- D_i is the relative deadline,
- P_i is the priority of the task,
- $U_i(t)$ is the task utility function, where t is the time elapsed since its arrival.
- $RR_i[]$ is resource requirement vector, which has the list of required and optional resources.
- L_i is the task criticality level,
- M_i is modality,
- $C_i(RA_i[])$ is the worst case execution time, which depends on resources currently allocated, $RA_i[]$, to the task.

Each resource R_x is defined as: $R_x = (n_x, t_x, sc_x, d_x[])$ where,

- n_x is the count of the available resources R_x,
- t_x is the resource type (Blocking/Non-blocking),
- sc_x is the worst case switching cost associated with the resource,

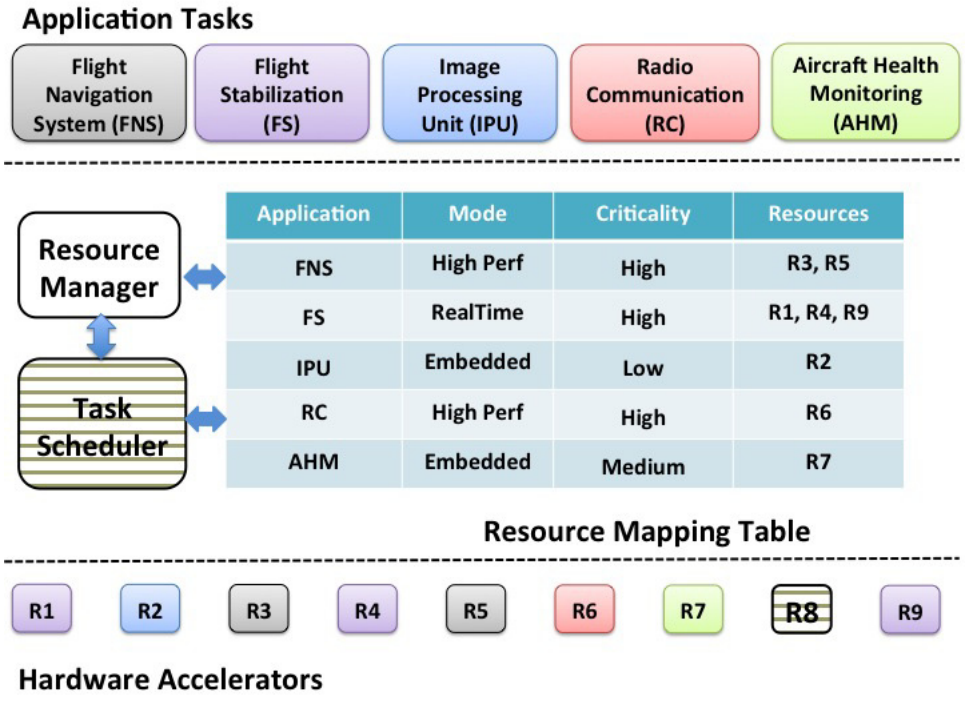

Fig. 5: Conceptual architecture of a resource manager and scheduler for the envisioned adaptive mixed criticality system.

- $d_x[]$ is the resource dependency vector.

One of our end goals is to develop a real-time resource manager and scheduler as part of the adaptive computational stack for heterogeneous mixed critical real-time systems, as conceptually illustrated in Figure 5. The hardware accelerators are dedicated hardware modules that are used to accelerate a certain functionality or operation. The applications make use of these hardware components to improve application throughput. Each application task is associated with a criticality level, which changes the resource requirements and parameters of the application. The criticality of applications may change during run-time, which is triggered by dynamically changing operating conditions (environment). The resource mapping table stores the task parameters, status and resource requirements of all applications. This is used by 1) the resource manager for the allocation and management of hardware resources, and 2) the task scheduler, which takes takes into account the criticality of the applications and generates schedules accordingly to ensure that the tasks are completed within their deadline. As the application parameters and resource requirements change during run-time, there is a need for a parametric-based on-line scheduling approach.

There are numerous challenges in scheduling and resource management for the envisioned adaptive mixed critical real-time systems. Some of the key challenges are:

- Identifying the stable states of execution and guaranteeing system stability during state transitions.
- Schedulability analysis: Scheduling space expands rapidly when there are n modes and n^2 transitions. The WCET of a task depends on the resources allocated and for n optional resources, there can be 2^n possible resource combinations for each task. Mitigating or overcoming this potentially overwhelming search space size is a key scheduling challenge.
- Defining the semantics of hardware accelerators.
- Calculating the speed-up factor for task using a hardware accelerator, if it is data dependent.

4. Conclusion

A vision for an adaptable computing platform to support DDDAS applications operating under unexpected conditions (i.e. *strategic surprise*) was introduced. We discussed the limitations of existing mixed criticality task models, which does not fully capture the dynamics of mixed-critical heterogeneous computing platforms and proposed an extended task model. We briefly discussed the challenges associated with developing scheduling and resource management algorithms for such a platform. These challenges are starting points for rich areas of continued research.

Acknowledgments

This work is supported in part by the National Science Foundation (NSF) under award CNS-1060337, and by the Air Force Office of Scientific Research (AFOSR) under award FA9550-11-1-0343.

References

[1] P. Lee, Colloquium presented at the University of Washington, http://www.youtube.com/watch?v=lLYobdHCCEo.
[2] S. Vestal, Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, in: Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International, 2007, pp. 239 –243.
[3] S. Baruah, S. Vestal, Schedulability analysis of sporadic tasks with multiple criticality specifications, in: Real-Time Systems, 2008. ECRTS '08. Euromicro Conference on, 2008.
[4] D. de Niz, K. Lakshmanan, R. Rajkumar, On the scheduling of mixed-criticality real-time task sets, in: Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE, 2009.
[5] K. Lakshmanan, D. de Niz, R. Rajkumar, G. Moreno, Resource allocation in distributed mixed-criticality cyber-physical systems, in: Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, 2010.
[6] N. Audsley, Optimal priority assignment and feasibility of static priority tasks with arbitrary start times, Citeseer, 1991.
[7] L. Sha, J. Lehoczky, R. Rajkumar, Solutions for some practical problems in prioritized preemptive scheduling, in: IEEE Real-Time Systems Symposium, 1986, pp. 181–191.
[8] S. Baruah, H. Li, L. Stougie, Towards the design of certifiable mixed-criticality systems, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE, 2010, pp. 13 –22.
[9] N. Guan, P. Ekberg, M. Stigge, W. Yi, Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems, in: Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd, 2011, pp. 13 –23.
[10] T. Park, S. Kim, Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems, in: Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on, 2011, pp. 253 –262.
[11] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, L. Stougie, Scheduling real-time mixed-criticality jobs, Computers, IEEE Transactions on 61 (8) (2012) 1140 –1152.
[12] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, L. Stougie, The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems, in: Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on, 2012, pp. 145 –154.
[13] K. Lakshmanan, D. de Niz, R. Rajkumar, Mixed-criticality task synchronization in zero-slack scheduling, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE, 2011.
[14] B. Huber, C. El Salloum, R. Obermaisser, A resource management framework for mixed-criticality embedded systems, in: Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE, 2008.
[15] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, L. Sha, Memory access control in multiprocessor for real-time systems with mixed criticality, in: Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on, 2012.
[16] C. Kumar, S. Vyas, J. A. Shidal, R. Cytron, C. Gill, J. Zambreno, P. H. Jones, Improving system predictability and performance via hardware accelerated data structures, Procedia Computer Science 9 (0) (2012) 1197 – 1205, proceedings of the International Conference on Computational Science, ICCS 2012.
URL http://www.sciencedirect.com/science/article/pii/S1877050912002505