Extending the RMS Paradigm of Model-Based Computing to Meet Requirements and Demands of Real-Time, Real-World Emergent Critical Processes

Martin J. Dudziak, PhD
TETRAD Technologies Group, Inc.
14.June.2007

Introduction

MPC and RMS

Model-based computing (MPC) is hardly a new concept in computing and arguably has its foundations in some of the early generations of artificial intelligence research going back to the 1970's and 1980's []. Qualitative physics [] and medical expert systems (e.g., Mycin []) are but two examples of model-based computing from an era of single-core microprocessors, LISP machines and RETI algorithms. One of the severe limitations of early AI and subsequent neural network and other statistics-based learning systems was the problem of tackling very large data sets in search and recognition algorithms; the need for trade-offs to accommodate processing limitations (including both processor speed and memory) brought a steep cost in meeting real-time demands or in reducing either false positives, false negatives, or both. The availability and affordability of computing resources that enable supercomputing capabilities to be applied to a large number of business and consumer applications now enables realistic MPC to be considered as more than an exercise for computer science researchers.

One of the more well-known representatives of MPC today (2007) is in the Recognition-Mining-Synthesis (RMS) paradigm advanced principally by Intel. RMS is an application software model for terascale computing and in particular the subject of research and development within Intel for its multi-core processor architecture []. The Intel Tera-scale architecture itself manifests a course of development in parallelism that can be traced to MIMD (multiple-instruction-multiple-data) platforms in early parallel processing that originated with the Inmos T-series transputer (T414, T800, T9000) and formally, even early, in the CSP (communicating sequential processing) models of C. A. R. Hoare and others [] in the early 1980s. Following a decade-plus love affair with shrinking photolithographic technology and increased processor speeds, the semiconductor industry began to address energy and performance efficiency by building from the foundations anew, with a change in scale such that parallelism, workload distribution, fault-tolerance (spares), local specific dedicated functions, and interconnect fabrics are now being down on-chip, on-die, rather than on-board. What goes around - gets smaller - then comes around all the same.

Today RMS shows great promise as the progenitor of a class of MPC architectures that can address a multiplicity of problems that have existed for a long time, if not forever, in the world, but which are dramatically demanding solutions which can only be assisted by high performance computing on a massively distributed and commonplace scale; i.e., within the computing world of homes, automobiles and public environments. This class of problem is one that is directly linked with the confluence of several factors and situations, ranging from the high human population on the planet and the increased affluence (esp. mechanization and mobility) of that population; energy demands and global climate change including a rise in nonlinear weather phenomena including major storms; an increase in the average lifespan, an increase in environmental hazards and pollutants, and a consequence increase in a variety of challenging; and a dramatic increase in local and global armed conflict including the easy facilitation of terrorism and acts of mass destruction. There are problems today that are more demanding of solutions that require massive data space searches, pattern matching trials and evaluations, interpolations and fittings among

complex n-dimensional objects, and synthetical reasoning, the complement of analysis and invariable a more complex process for computational systems because of the open-ended paths that need to be explored in order to determine some optimal conclusion, whether that be in face recognition, obstacle avoidance, molecular dynamics, or personal video management. Many of these problems are extraordinarily "critical" for the lives of millions, either because the problem is something that can affect large numbers of peoples' lives at once (e.g., situations concerning climate change, pandemics or terrorism), or because many face the same issues (e.g., traffic safety or individual health maintenance). Others are less critical in the sense of life-or-death but are perceived as demands and needs by a society of consumers that has become accustomed to living with increasing masses of personal data such as audio and video, all of which needs the power offered by an RMS model of computing in order to satisfy the individual consumer. The attention in this brief paper, however, will be upon the former class of such critical problems, namely those that can make differences of life or death for large numbers of people in their individual lives or en masse together.

ECP

The term for this class of problems is Emergent Critical Process or ECP (also referred to as Emergent Critical Event or ECE). These are inherently nonlinear and catastrophic from a mathematical perspective, and generally they have the potential of being catastrophic on physical, social, and economic levels when they occur and especially when they occur in manners for which the subject population is unaware and unprepared. ECP problems may be modeled as simple bifurcations and two-dimensional flows, or as s-called strange attractors (e.g., Lorenrz, Rössler, Tamari, Hénon) or more likely in mass-social settings, the model will be itself composite and involving interactions among a set of different behaviors, some of which may be quite linear and others frankly non-algorithmic and requiring good heuristic judgment for approximations. The December, 2004 Indian Ocean Tsunami is one case in point. Almost any earthquake is another. Hurricane Katrina, albeit an ECE with much forewarning and plenty of computer simulations and predictions for its coming, was a classic ECP. Intentionally-triggered events such as the 9-11 WTC attack is yet another type of ECP. In all these cases the ECP is not "simply" the physical event such as a hurricane, tornado, earthquake or explosion that can be modeled in reasonable fashion well enough, but a far larger set of events that involve massive numbers of components such as individual persons, vehicles, and systems which are interacting and with a variety of dependency relationships, not all of which can be expected to be evident early-on in the modeling process even for subject experts. This means that the element of discovery (novelty, innovation) is something that must be considered in many computational approaches to managing ECP, and this is a dimension of the classic AI (or simply "I" for Intelligence) problem that needs to be addressed in thinking about paradigms for model-based computing.

ECP and specific ECE are not, however, limited to massive newsbreaker phenomena that affect thousands or millions of people at once. They may be extremely localized in scale and in type of effect upon people's lives. These are ECP in the home, in the automobile, in the body, in everyday affairs of life. There are many ECP that happen every day and night, and these are the events that provide a very large and open-ended set of application opportunities by which hardware and software developers, and the developers of many consumer products, can and should take advantage of MPC and RMS-type models in particular. These are also the types of ECP computing challenges that, if intelligently addressed for not only processing capabilities and programming methods but also for human-machine interfaces and the proverbial "user friendliness" will open up massive opportunities for high performance computing (HPC) in the general consumer world.

ECP that occur in the home, on the road, in the community, and in the body are the subject of this brief memo wherein are presented some modifications to the RMS paradigm and some suggestions for generalizing MPC. The objective of making some changes and creating an abstract MPC is to create a set of problem-solving methods and tools that can be employed in use-case modeling and code generation for use in a very wide range of applications, for using such processors as the Tera-scale devices engineered by Intel, without forcing developers to continually reinvent basic techniques and code. The complexities of programming an 8-core, 80-core or larger single-die system are severe and demanding for most programmers. The complexities of the applications are even more demanding, especially if one considers that by virtue of the large number of input devices or data streams, data mining load, and synthesis outcomes, there is likely to be a frequent need to add, subtract, and modify processes that must be performed in the RMS type of application. Unless the overall application can easily accommodate modification and retrofitting of new input, processing and output types, the computational values and the economic benefits associated with Tera-scale computing will be significantly cut back by losses in optimal performance and costs in modifications to the application software. Lest the system be cast back into the type of problems that beset mainframe computing and pre-object-oriented software development of the 1970's and 1980's, it will be important to develop an abstract RMS application development tool that can be used easily for building new systems and for modifying existing applications. The paradigm is something like but not exactly like the APIs and database query tools that have evolved in conventional applications software.

Completing RMS → DRMSO

The model of RMS is effective but perhaps a bit incomplete and this needs to be addressed briefly before proceeding. Recognition-Mining-Synthesis is only part of the picture, a very central part, but it omits two important process components that cannot be easily lumped together into this model. Ahead of recognition there is Detection. It can be called, variously, sensing, data collection, observation, noticing, gathering, or triggering. It is the process, potentially very computationally intensive, of finding that-which-is-to-be-recognized, the pattern that is identifiable as a pattern, or as an anomaly, or as a trigger, but which is not yet recognized or classified. Examples will help to clarify this process as well as its relationship with Recognition and its importance for Mining, Synthesis, and the second mostly-omitted process component, Optimization. After Synthesis, there is generally, in most applications, not a final static state, but an iteration or a continuation or the Synthesis process with new information that may be a return to some part of the Detection or Recognition phases. Often there is a need, or at least an appropriate place, for optimizing that which has been synthesized and which is being put into action as a response, an output, a final throughput from the rest of the process pipeline. Again, it is hopes that concrete examples will help in clarification and in understanding relationships.

[[[balance is the subject of current research and a possible theme for presentation, contract work and teaching by the author]]]