Integrated Hardware and Software for No-Loss Computing

Computations on parallel processors can continue, even if one processor fails.

NASA's Jet Propulsion Laboratory, Pasadena, California

When an algorithm is distributed across multiple threads executing on many distinct processors, a loss of one of those threads or processors can potentially result in the total loss of all the incremental results up to that point. When implementation is massively hardware distributed, then the probability of a hardware failure during the course of a long execution is potentially high. Traditionally, this problem has been addressed by establishing checkpoints where the current state of some or part of the execution is saved. Then in the event of a failure, this state information can be used to recompute that point in the execution and resume the computation from that point.

A serious problem arises when one distributes a problem across multiple threads and physical processors is that one increases the likelihood of the algorithm failing due to no fault of the scientist but as a result of hardware faults coupled with operating system problems. With good reason, scientists expect their computing tools to serve them and not the other way around.

What is novel here is a unique combination of hardware and software that reformulates an application into monolithic structure that can be monitored in real-time and dynamically reconfigured in the event of a failure.

This unique reformulation of hardware and software will provide advanced aeronautical technologies to meet the challenges of next-generation systems in aviation, for civilian and scientific purposes, in our atmosphere and in atmospheres of other worlds. In particular, with respect to NASA's manned flight to Mars, this technology addresses the critical requirements for improving safety and increasing reliability of manned spacecraft.

This work was done by Mark James of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*

In accordance with Public Law 96-517, the contractor has elected to retain title to this invention. Inquiries concerning rights for its commercial use should be addressed to:

Innovative Technology Assets Management
JPL
Mail Stop 202-233
4800 Oak Grove Drive
Pasadena, CA 91109-8099
(818) 354-2240
E-mail: iaoffice@jpl.nasa.gov

Refer to NPO-42554, volume and number of this NASA Tech Briefs *issue, and the page number.*

Decision-Tree Formulation With Order-1 Lateral Execution

Some decision trees can be transformed into objects executable by simple table lookups.

NASA's Jet Propulsion Laboratory, Pasadena, California

A compact symbolic formulation enables mapping of an arbitrarily complex decision tree of a certain type into a highly computationally efficient multidimensional software object. The type of decision trees to which this formulation applies is that known in the art as the Boolean class of balanced decision trees. Parallel lateral slices of an object created by means of this formulation can be executed in constant time — considerably less time than would otherwise be required.

Decision trees of various forms are incorporated into almost all large software systems. A decision tree is a way of hierarchically solving a problem, proceeding through a set of true/false responses to a conclusion. By definition, a decision tree has a treelike structure, wherein each internal node denotes a test on an attribute, each branch from an internal node represents an outcome of a test, and leaf nodes represent classes or class distributions that, in turn represent possible conclusions. The drawback of decision trees is that execution of them can be computationally expensive (and, hence, time-consuming) because each non-leaf node must be examined to determine whether to progress deeper into a tree structure or to examine an alternative. The present formulation was conceived as an efficient means of representing a decision tree and executing it in as little time as possible.

The formulation involves the use of a set of symbolic algorithms to transform a decision tree into a multi-dimensional object, the rank of which equals the number of lateral non-leaf nodes. The tree can then be executed in constant time by means of an order-one table lookup. The sequence of operations performed by the algorithms is summarized as follows:

1. Determination of whether the tree under consideration can be encoded by means of this formulation.
2. Extraction of decision variables.
3. Symbolic optimization of the decision tree to minimize its form.
4. Expansion and transformation of all nested conjunctive-disjunctive paths to a flattened conjunctive form composed only of equality checks when possible.

If each reduced conjunctive form contains only equality checks and all of these forms use the same variables, then the decision tree can be reduced to an order-one operation through a table lookup. The speedup to order one is accomplished by distributing each decision variable over a surface of a multidimensional object by mapping the equality constant to an index.