DISTRIBUTED PERFORMANCE MAINTENANCE

William Bricken
November 1984

[A synopsis of work I did with colleagues in modeling performance failures in distributed systems. My role was primarily that of implementation.]

Distributed processing poses several difficult technical problems. A Performance Maintenance System (PMS) that monitors the functioning of the nodes in a distributed system is an integral component of a distributed configuration, for these reasons:

1. The PMS can identify failures and potential failures at a node of the distributed system.

2. The PMS can initiate recovery methods to restore partial functionality.

3. The PMS furnishes design information that contributes to the achievement of proper distribution of system resources.

Our approach was to design a general architecture for Artificial Intelligence (AI) based distributed system control, and to construct a simulation of a PMS with limited capabilities within this general architecture. The architecture, which serves as the modeling environment, demonstrated real-time functionality as an intelligent, distributed operating system. The PMS implemented within the architectural framework demonstrated a working prototype of failure detection, and of (limited) diagnosis and recovery techniques for controlling performance of a simulated distributed computation system.

EXPANSION OF THE SIMULATION

The progress we have made in the characterization and implementation of faults associated with intermittent hard failures of nodes has been at the design level. The primary problem is the incorporation of a representation of time into a model of a distributed system. An intermittent failure must be communicated to other nodes as a state variation over time. Consequently this effort evolved into a study of the questions associated with the representation of time, both locally and globally. We are currently considering the modeling of local time frames that are not necessarily coordinated globally. Another approach we are considering is the treatment of time as a function rather than a parameter. The representation of time is a classically difficult problem in AI, and the constraints of the interim funding period did not allow sufficient allocation for a full study of this problem.

For the specification of intermittent hard failures, we recognized that we needed to construct design tools to handle experimentation with time variables. Some effort has been focused on the design of these prototyping tools. One promising approach is the use of model-based representation and reasoning techniques. A node in a distributed system is represented by a series of models that represent evolution over time. Thus, the reasoning system has access to a series of representations (models) of a particular node, some of which may be characterized as functioning, others of which may be failed. Since the other nodes communicate with only one model at a particular time, the functional state of that model can vary without introducing interactions into the system. In effect, intermittent failure is modeled by the change of a state variable between two different times of model construction.

ADDITIONAL CONTROL TECHNIQUES

The techniques we explored can be grouped under the concept of Qualitative Model-Based Reasoning (QMBR). Basically, the model-based approach focuses the inference capabilities of a system on a collection of facts that provide a model of the behavior of a node in addition to individual descriptive parameters associated with that node. We anticipate that this approach will have several advantages:

1. Models of a node would serve as the representational and inferential basis for decisions about inter-node communication.

2. Nodes can be modeled as composite (modular) components, that can be viewed or addressed at various levels of detail.

3. Messages between nodes can be modularized and lumped, decreasing the number and the frequency of inter-node communications.

4. Fault states can be represented abstractly, permitting both an economy of representation, and an appropriate grain-size (abstraction level) from which to characterize the fault.

5. Different models or perspectives can be represented. Thus, for example, we could view a node from a logic-circuitry model or from a communication-transmission model, specifying different functionalities depending on the type of model.

Our initial investigation of this technique has been in a controlled testbed environment on prototype hardware diagnostic problems (such as logic circuitry).

The selection of an appropriate model

There is no single model of reality that is suitable for all contexts. How a component is modeled depends on the set of questions that are to be answered about that model. To aggravate matters, the possible models are not necessarily independent of each other. Our research in this area has accentuated the care with which one must align the model with the intended purpose. For fault identification and diagnosis, a taxonomy of the expected faults is critical. Models designed to identify unbalanced processor loads at a particular time, for example, may be poor for identifying time-varying behaviors.

A related problem is the identification of the correct level of abstraction for the analysis of a particular fault. With the QMBR approach, a system can be modeled by a hierarchy of models, each level of the hierarchy representing a different level of qualitative abstraction or detail. Generally, a fault modeled at a given level will be represented at all the more detailed levels. However, the expression of a fault at too refined a level of detail leads to a clumsy and computational expensive representation, with a resulting degradation of the system's ability to succinctly identify what to do to correct that fault. On the other hand, failure to model a fault state at a sufficient level of detail may result in an inability to properly localize the source of the failure. The correct level of abstraction from which to characterize a particular fault is critical; identifying this level appears to be a domain dependent problem.

The complexity of representation and search

Closely allied with the question of appropriate models is the combinatorial explosion associated with modeling fault states. We have encountered this phenomenon in our prototype diagnostic problems in two different forms:

a. The number of fault states to be modeled. Even in toy environments, the number of ways things can go wrong is very large.

b. The number of hypotheses about specific fault states. The time evolution of a fault state has a large branching factor. It is difficult to know what states a fault state will change to.

These complexity problems are common in AI reasoning. Our preliminary efforts to develop a diagnostic technique based on QMBR hold great promise in addressing such problems. In the Phase II effort we will extend this work on QMBR as the basis for our real-time diagnostic and performance maintenance inference techniques.

MANAGEMENT OF INDIVIDUAL NODES

The intent of this task was to design and incorporate management and control specialist Knowledge Sources (KSs) within the control structure of each simulated node. These KSs would trigger whenever specific events occurred; in turn, they would effect the control of node activity. This task is design intensive, since it is necessary to identify specifically the effects on system management of each computational context. We studied the design of control regimes for these contexts which a node might encounter:

a. load out of balance (compared to other nodes),

b. incoming task assignments (from other superior nodes) and outgoing task assignment (to inferior nodes),

c. some simple types of failure at a node (total failure, goal abandonment due to overload, goal abandonment due to insufficient time for completion of task), and

d. recovery processes for the failure types studied.

Preliminary results indicate that distribution of some control and management responsibilities to individual nodes is both desirable and achievable. Generally, the division of responsibility between a global control mechanism and local node control mechanisms is context dependent. We are working in the direction of maximizing local control, and expect to be able to transfer most global control mechanisms to local nodes. The responsibility usually invested in the global management system will be carried by the inter-node communication protocols. One promising avenue of research involves local KSs that alter the priority of triggered control actions on the basis of changing context. Thus, for example, if a node experiences a heavy task loading, local KSs would evaluate the agenda of triggered actions, and reprioritize them to maximize the expected completion of critical tasks. Another approach involves the reestimation of probabilities associated with the hypotheses that a node keeps about its current working environment. The probabilistic credibility of a hypothesis is locally recomputed, for example, as a function of incoming messages. This, in turn, leads to local control decisions impacting the use of that hypothesis structure as control data.

Areas of focus for the integration of PMU hardware design with our software simulation include:

1. the set of messages necessary to convey fault identification information,
2. the synchronization of timing and control across the nodes,
3. the PMU monitoring protocols for error messages,
4. the task-oriented message vocabulary, and
5. the amount of message traffic necessary for fault diagnosis.

SUMMARY

Three specific tasks were addressed:

TASK 1: Expansion of the simulation to include intermittent hard failures.

Design of fault models for intermittent hard failures raised questions about the representation of time variables in the PMU simulation. Designs involving time were studied but not implemented.

TASK 2: Qualitative model-based reasoning control techniques.

Prototype diagnostic problems were solved using the QMBR representation technique. Issues studied in depth were

a. the selection of the appropriate model, and
b. the complexity of representation and search.

Hierarchical levels of model abstraction was shown to be both a powerful and useful representation technique.

TASK 3: Architecture integration.

We implemented a major redesign of the architecture, which included enhanced uniformity and modularity of representation, and the development of rapid prototyping tools.