Challenges in Engineering Distributed Shipboard Control Systems

Lonnie R. Welch and Binoy Ravindran
Computer Science and Engineering Dept.
The University of Texas at Arlington
Box 19015
Arlington, TX 76019-0015
welch@cse.uta.edu

Robert D. Harrison, Leslie Madden, Michael W. Masters and Wayne Mills
Code B35
Naval Surface Warfare Center
Dahlgren, VA 22448

Abstract

This paper presents a representative Navy shipboard control system and discusses its distributed implementation. The real-time, fault tolerance and scalability aspects are discussed; the system configuration is described; and the computing infrastructure is summarized. The paper also presents challenges which are likely to be faced by the engineers of the next generation of Navy surface combatants.

1. Introduction

Although the end of the Cold War minimized the threat of massive open ocean naval combat, new littoral warfare missions and scenarios involving both combined arms and joint allied operations, have emerged. New threats, missions and operating environments present a greater challenge to target identification, reaction time, command and control, and tracking and weapons accuracy, requiring even more computer automation and data processing than in the past. As new systems designed to meet these challenges enter fleet service, it is increasingly difficult to engineer cost effective computing systems with traditional military standard computing resources. The expected growth in required computing capacity, and the architecture, based on Navy standard computers and architecturally limiting point-to-point interfaces, lacks the scalability needed to add the required new capacity. A longer term, cost effective solution, based on full use of commercial computing resources and capable of being expanded over a distributed pool of resources as requirements grow, is needed.

In response to the need to develop high capacity, scalable computer systems for shipboard use, a program called the High Performance Distributed Computing Program (HiPer-D), was created. HiPer-D is intended to provide the technical design concepts and engineering data needed to enable the Navy to capitalize on commercial computing products. The program, conducted jointly by the Defense Advanced Research Projects Agency (DARPA) and the Aegis Shipbuilding Program, consists of simultaneous top down engineering studies and large scale critical experiments using new computer technology.

HiPer-D is intended to validate distributed processing concepts not only for future Aegis ships but also for a new class of surface combatant ships, presently projected to begin construction in the year 2003. This ship, currently referred to as the Surface Combatant - 21st Century, or simply as SC-21, is intended to be a substantial departure from previous ships, and even future Aegis ships, with regard to computer technology. A key anticipated future direction is expansion of the role of computing to encompass not only combat system functionality but also ship systems and perhaps administrative tasks as well. Eventually, the ship's computing resources...
may be viewed in the same way that traditional hull, mechanical and electrical resources are viewed today, i.e., as part of the ship's infrastructure or "hotel services." Computing resources such as these would provide a genuinely new option for ship automation, bringing to reality the idea of "ubiquitous computing," that is, the ability to compute wherever and whenever needed.

The aim of the HiPer-D program, and the entire Aegis advanced baseline engineering process, in supporting the Aegis transition from military standard computing to COTS-based (COTS is an acronym meaning commercial off-the-shelf) computing has been to mitigate the risks associated with the major changes that will be required. To that end, a substantial portion of the HiPer-D investment focused on examining the available computing technologies and devising an evolutionary approach that would permit controlled and managed change to the combat system to occur without introducing undue cost and schedule risk into the baseline engineering process. The result of this study is to adopt a strategy of evolutionary transition toward a very advanced, fully distributed architecture that might supply the entire shipboard computing needs of future surface combatants.

The purpose of this paper is to present a representative Navy shipboard control system, to discuss its distributed implementation by the HiPer-D team, and to present infrastructure support for such a system. Section 2 provides an overview of the shipboard anti-air-warfare (AAW) control system which has been implemented within the HiPer-D Testbed. Section 3 describes a distributed implementation of the AAW system. Related work is discussed in section 4, and conclusions and future work are stated in section 5.

2. A shipboard AAW software system

Figure 1 depicts the major components of the HiPer-D Testbed application (a subset of an AAW system), and categorizes them as sensors, filter and sense software elements, evaluate and decide software elements, act software elements, and actuators. The Testbed implementation of the AAW system includes the following capabilities: a simulated track source; track correlation and filtering algorithms; track data distribution services; a doctrine server and three types of doctrine processing (semi-auto, auto-sm, and auto-special); an engagement server; a display subsystem including X-windows based tactical displays, submode mediation, and alert routing; surface operations (reengineered from CMS-2 to Ada), and simulated weapons control system and identification upgrade capabilities.

3. A distributed implementation of the AAW software system

The current HiPer-D Testbed contains a distributed system providing functionality to support a standard missile path through the combat system. It supports up to

<table>
<thead>
<tr>
<th>PROCESSOR (QUANTITY)</th>
<th>OPER. SYS.</th>
<th>NETWORKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun Ultra 4000, 6 processor SMP (1)</td>
<td>Solaris 2.5.1</td>
<td>Ethernet/FDDI/Myrinet</td>
</tr>
<tr>
<td>Sun Server 1000, 8 processor SMP (1)</td>
<td>Solaris 2.5.1</td>
<td>Ethernet/FDDI/ATM</td>
</tr>
<tr>
<td>Sun Ultra 1 (2)</td>
<td>Solaris 2.5.1</td>
<td>Ethernet/FDDI/Myrinet</td>
</tr>
<tr>
<td>Sun Ultra 2, 2 processor SMP (1)</td>
<td>Solaris 2.5.1</td>
<td>Ethernet/FDDI/Myrinet/ATM</td>
</tr>
<tr>
<td>Pentium 90Mhz (6)</td>
<td>OSF RT</td>
<td>Ethernet/FDDI/Myrinet</td>
</tr>
<tr>
<td>HP TAC-4 J210 (7)</td>
<td>OSF PA RT or HPUX 10.10</td>
<td>Ethernet/FDDI/Myrinet or Ethernet/FDDI/ATM</td>
</tr>
<tr>
<td>Alpha 2100 4 processor SMP (1)</td>
<td>DEC Unix 3.2</td>
<td>Ethernet/FDDI</td>
</tr>
<tr>
<td>Alpha 3000 (4)</td>
<td>DEC Unix 3.2</td>
<td>Ethernet/FDDI/ATM</td>
</tr>
<tr>
<td>Alpha 200 (5)</td>
<td>DEC Unix 3.2</td>
<td>Ethernet/FDDI</td>
</tr>
</tbody>
</table>

Table 1. The HiPer-D hardware platform.
2000 tracks with up to 15 engagements in progress. It consists of more than 50 processes (not including COTS components). It has both fault-tolerant and scalable components. The Testbed currently runs over the heterogeneous network configuration described in Table 1. In addition to supporting Aegis functionality, the HiPer-D Testbed has facilities to support real-time performance graphics monitoring and display, and post-mortem performance analysis.

These facilities are used to dynamically monitor system performance, to compare performance against the current Aegis Weapon System, and to support dynamic management of resources.

The real-time, fault tolerance and scalability paradigms employed within HiPer-D’s distributed implementation of the AAW system are illustrated in Figure 2. Real-time constraints are stated with respect to control paths which span one or more elements of the control system. Figure 2 depicts a time constrained path that begins with two sensors; the sensor data feed two filter elements, which provide input to an evaluate and decide element; the evaluate and decide element triggers two act elements, which control a single actuator. Such an execution path is typically the unit for which timing requirements are specified and assessed in the HiPer-D project. The large granularity at which timing requirements are specified has ramifications on the granularity with which resource allocation and timing assessment are performed. In addition to referring to large grain entities, timing characteristics for the AAW system include: dynamically changing periods; sporadically arriving, transient periodic tasks with dynamically determined periods; tactical loads which vary by several orders of magnitude; unknown upper bounds on required processing; adaptation to missed timing requirements; and high priority sporadics.

Application-level fault tolerance is achieved by replication of tactical programs (see figure 2), resulting in a primary (active) copy and secondary (passive) copies residing on different processors. Current state is maintained by all copies of a program. When a replica terminates abnormally, all replicas detect this automatically (through notification from middleware services). If the primary replica terminates, a new primary is selected automatically (by the middleware). This results in virtually no loss of functionality, as switchover to the new primary occurs very quickly. To maintain survivability, a new replica is automatically restarted by the resource manager. The determination of the location at which to restart the replica involves prediction of quality of service. State is transferred to the new replica automatically (by the middleware).

Replication of tactical programs is also employed to achieve application-level scalability (see figure 2). Programs are made scalable by sharing tactical load among replicas. Individual replicas self-schedule their own portion of the tactical load, by considering the number of replicas currently in the system. Since replicas automatically detect when one replica dies, self-scheduling can adapt to varying numbers of replicas. When a load-sharing replica terminates abnormally, a new replica is automatically restarted by the resource manager.

Figure 3 depicts a computational infrastructure that is appropriate for large grain, distributed control systems. A system engineer specifies the characteristics of the time-constrained paths which constitute application software and also specifies the characteristics of the distributed hardware. Real-time instrumentation provides cornerstone metrics which characterize such things as
the performance and resource usage requirements of the paths. The information gathered during instrumentation is used by a dynamic resource manager to make allocation decisions, and is provided to system visualization components which provide various views of the quality of service being provided to the application system. HiPer-D has developed a computational infrastructure that contains prototypes of the major components depicted in Figure 3.

4. Related work

Resource management in real-time systems has extensively been addressed (e.g., see [2,3]). The models defined in most of these efforts consider tasks with timing constraints imposed on them. Further, most of the efforts are based on worst case assumptions regarding execution times and resource usage.

In contrast, systems such as the distributed AAW system consist of multitudes of programs that communicate in an irregular, non-trivial manner, with each program having a hierarchical software structure. When engineering such systems, it is useful to specify timing constraints for very large grain entities (e.g., paths). A path may have large variability in its execution time (e.g., 3 orders of magnitude!). Also, many path-based systems have dynamically changing periods, high priority aperiodics, and scalability and dependability requirements. As stated by Koob in [1], the application of techniques which are designed to handle anticipated peak demands leads to (1) inefficient resource utilization and (2) inadequate crisis response. The engineering of systems having the dynamic, large grain characteristics of the AAW system present many challenges to technologists. Can the existing paradigms [2,3] for real-time computing be extended to meet the challenges? Is a new paradigm needed? The answers to these questions will become clear as systems such as the Navy’s 21st Century Surface Combatant are engineered.

5. Conclusions

This paper describes a distributed implementation of a representative shipboard control system. The distributed implementation has been successfully engineered by the HiPer-D team using primarily commercial technology. The HiPer-D prototype incorporates many critical components of the Aegis Weapon System, and thus provides a platform for experimental validation of concepts. An initial computational infrastructure has been engineered by the HiPer-D team, and is employed to provide real-time, fault tolerance, and scalability services to the distributed AAW application. Ongoing work includes evolution of the infrastructure to address (1) quality of service assessment, (2) adaptive resource management, and (3) systematic study of HiPer-D’s distributed AAW system to produce mathematical characterizations and a distributed shipboard computing benchmark suite.

6. References


5. Acknowledgements

The authors would like to acknowledge the generous support of the Naval Surface Warfare Center and the Defense Advanced Projects Agency.