

AFRL-IF-RS-TR-2002-185
Final Technical Report
August 2002



REAL-TIME APPLICATION PERFORMANCE STEERING AND ADAPTIVE CONTROL

University of Illinois

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. D516

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-185 has been reviewed and is approved for publication

APPROVED:

A handwritten signature in black ink, appearing to read "Edward Depalma", with a long horizontal flourish extending to the right.

EDWARD DEPALMA
Project Engineer

A handwritten signature in black ink, appearing to read "Michael L. Talbert", with a stylized, looped structure.

FOR THE DIRECTOR:

MICHAEL L. TALBERT, Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE AUGUST 2002		3. REPORT TYPE AND DATES COVERED Final Jul 96 – Jun 99
4. TITLE AND SUBTITLE REAL-TIME APPLICATION PERFORMANCE STEERING AND ADAPTIVE CONTROL			5. FUNDING NUMBERS C - F30602-96-C-0161 PE - 62301E PR - H767 TA - 00 WU - 11	
6. AUTHOR(S) Daniel A. Reed				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Illinois Department of Computer Science Urbana Illinois 61801			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Project Agency AFRL/ITB 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-185	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Edward DePalma/ITB/(315) 330-3069/ Edward.DePalma@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) High-performance computing is rapidly expanding from single parallel systems to distributed collections of heterogeneous sequential and parallel systems. The emerging applications are irregular, with complex, data dependent execution behavior, and dynamic, with time varying resource demands. The objective of the Real-time Application Performance Steering and Adaptive Control project is to replace ad hoc, post-mortem performance optimization with an extensible, portable, and distributed software infrastructure for real-time adaptive control that dynamically optimizes the performance of distributed applications. By integrating dynamic performance instrumentation and on-the-fly performance data reduction with configurable, malleable resource management algorithms and a real-time adaptive control mechanism, flexible runtime systems could automatically choose and configure resource management algorithms based on application request patterns and observed system performance. Such an adaptive resource management infrastructure can increase portability by allowing application and runtime libraries to adapt to disparate hardware and software platforms and increases achieved performance by choosing and configuring those resource management algorithms best matched to temporally varying application behavior. The Autopilot real-time adaptive control infrastructure is based on this thesis. Autopilot provides a flexible set of performance sensors, decision procedures, and policy actuators to realize adaptive control of applications and resource management policies on both parallel and wide area distributed systems.				
14. SUBJECT TERMS AUOPILOT, Real-Time Adaptive Control, Resource Management Algorithms			15. NUMBER OF PAGES 9	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Contents

1	Overview.....	1
2	Project Approach.....	1
3	Autopilot Software Overview	3
4	Project Accomplishments and Impact	4
5	Software Availability	5

1 Overview

The scope of high-performance computing is rapidly expanding from single parallel systems to distributed collections of heterogeneous sequential and parallel systems. Moreover, emerging applications are irregular, with complex, data dependent execution behavior, and dynamic, with time varying resource demands. In consequence, application developers increasingly complain that even small changes in application structure can lead to large changes in observed performance.

The performance sensitivity of current parallel and distributed systems is a direct consequence of resource interaction complexity and the failure to recognize that resource allocation and management must evolve with applications, becoming more flexible and resilient to changing resource availability and resource demands. Currently, software developers are forced to engage in a time consuming cycle of program development, performance measurement, and tuning to create non-portable code that conforms to parallel and distributed system idiosyncrasies.

Distressingly, the space of possible performance optimizations is large and non-convex, and the best match of application and resource management technique is seldom obvious *a priori*. Performance instrumentation and analysis provide the data necessary to understand the causes for poor performance *a posteriori*, but alone they are insufficient to adapt to temporally varying application resource demands and systems responses. Because the interactions between application and system software change across applications and during a single application's execution, we believe runtime libraries and resource management policies are needed that can adapt to rapidly changing application behavior.

By integrating dynamic performance instrumentation and on-the-fly performance data reduction with configurable, malleable resource management algorithms and a real-time adaptive control mechanism, flexible runtime systems could automatically choose and configure resource management algorithms based on application request patterns and observed system performance. Such an adaptive resource management infrastructure can increase portability by allowing application and runtime libraries to adapt to disparate hardware and software platforms and increases achieved performance by choosing and configuring those resource management algorithms best matched to temporally varying application behavior.

Based on this thesis, we developed *Autopilot*, a real-time adaptive control infrastructure. As described below, *Autopilot* provides a flexible set of performance sensors, decision procedures, and policy actuators to realize adaptive control of applications and resource management policies on both parallel and wide area distributed systems (computational grids).

2 Project Approach

Emerging defense and civilian applications are parallel, distributed, and mobile, are driven by real-time data sources, have time varying resource demands, and must accommodate dynamically changing resource availability (e.g., due to failures or resource contention). At present, optimizing the performance of these applications

requires multiple iterations of *ad hoc* measurement, post-mortem analysis, and platform-specific tuning, limiting performance, resilience, portability, and adaptability.

The objective of this project is to replace *ad hoc*, post-mortem performance optimization with an extensible, portable, and distributed software infrastructure for real-time adaptive control that dynamically optimizes the performance of distributed applications. Via this cross-platform optimization infrastructure, developers can build robust, distributed, mobile applications that are resilient to changing resource availability.

To support adaptive control, we built and validated a C++ infrastructure called *Autopilot* that can be used to create nimble applications that can dynamically reconfigure their behavior to meet changing resource conditions. *Autopilot* embodies the following features:

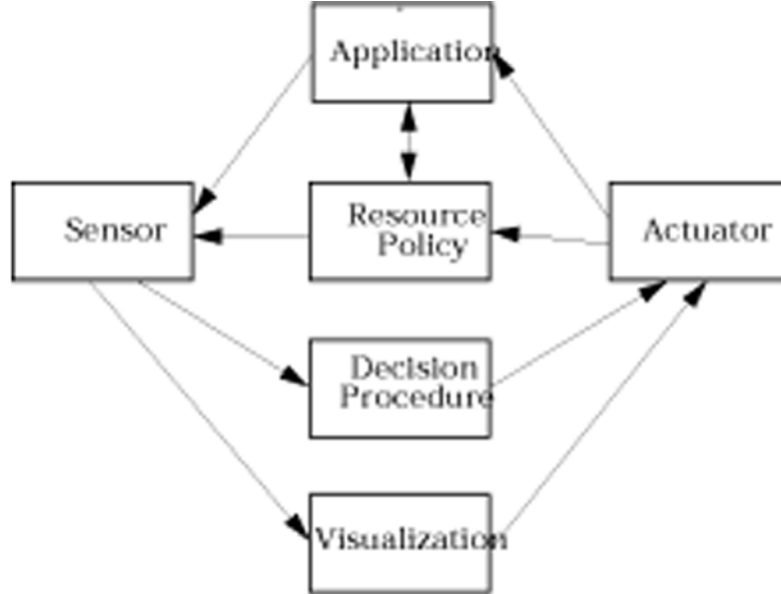
- *Distributed performance sensors* that can capture application and system performance data and generate performance metrics.
- *Software actuators* that can enable and configure application behavior and resource management policies.
- *Decision procedures* for selecting resource management policies and enabling actuators based on observed application resource requests and the system responses captured by performance sensors.
- *Distributed name servers* that support registration by remote sensors and actuators and property-based requests for sensors and actuators by remote clients.
- *Sensor and actuator clients* that interact with remote sensors and actuators, monitoring sensor data and issuing commands to actuators.
- *Desktop performance visualization tools* to provide analysts insight into the interaction of application demands and resource management algorithm response.

In this design, performance instrumentation sensors capture and compute quantitative application and system performance metrics. This data is used by decision procedures to choose and configure resource management policies via software actuators.

We tested the *Autopilot* infrastructure by developing an adaptive, high-performance I/O toolkit called PPFSII¹. Our earlier measurements of application I/O patterns and file system responses showed that achievable performance was strongly sensitive to small changes in either access patterns or policies. By using *Autopilot* sensors to measure I/O patterns, together with fuzzy logic rules for file system policy selection and actuators for policy configuration, PPFS II yielded dramatic improvements to I/O performance.

We also validated *Autopilot* using a large-scale parallel application. This application, developed by the DOE ASCI Center for the Simulation of Advanced Rockets (CSAR), is written in Fortran 90 using MPI. Although the solid rocket burn requires only two minutes, simulating 0.5 seconds of the burn is estimated to require 200 hours on a 128-

¹ Primary funding for PPFS II development was from other sources.



node SGI Origin2000. With the high cost of rocket failures, optimizing this simulation is of great practical importance.

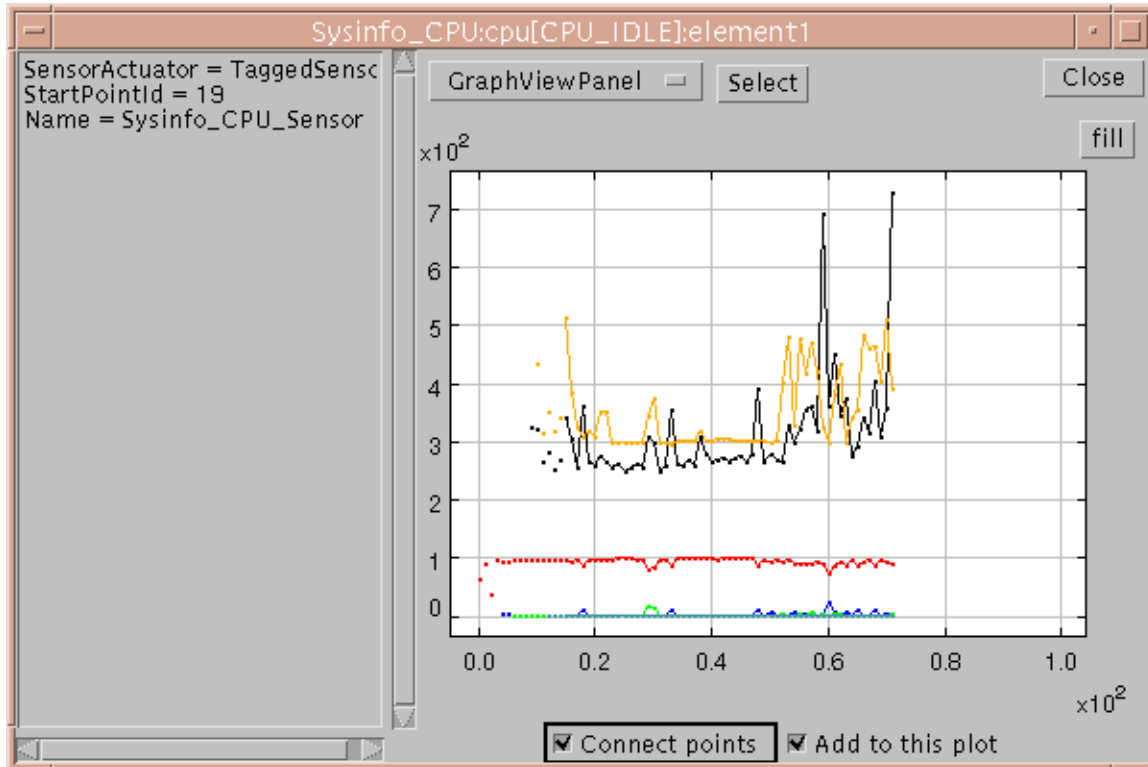
We instrumented the CSAR code by inserting *Autopilot* sensors that compute (a) the number of invocations and time spent in each procedure of the code's call graph and (b) capture data on the execution of logical code regions (e.g., initialization, fluids, solids, and output). We then combined this sensor data with actuators to control remote performance data capture in real-time. This integrated, real-time performance visualization and control experiment represents a major validation of *Autopilot* and provides functionality not present in other distributed/parallel performance measurement toolkits.

3 Autopilot Software Overview

Autopilot is implemented as a set of C++ classes and is built atop the DARPA-funded Globus wide-area computing toolkit. Globus provides a shared address space across local and wide-area networks and enables interprocess, intraprocess, and intermachine data sharing. Globus also supports heterogeneity, allowing a single computation to use multiple communication protocols, executables, and programming models.

Using Globus as a base, the *Autopilot* toolkit defines sensors, actuators, decision procedures, and sensor/actuator managers, all accessible via Globus "global pointers:" see Figure 1. Sensors are low-overhead routines designed to capture real-time performance data from distributed software components and can be extended via user-defined functions to process raw performance data before transmission (e.g., computing a profile from event trace data). In turn, actuators define a mechanism for implementing control functions in software modules (e.g., changing policies or policy parameters in

response to remote decision procedures). Sensor/actuator managers can be viewed as



access points for sensors and actuators.

Decision procedures implement fuzzy logic control of distributed software. They accept and evaluate real-time data from one or more sensors and generate actuator outputs in response. Our fuzzy logic toolkit is based on a subset of the publicly available ComNets Class Library (CNCL), with modifications and extensions for software control.

Using sensors and actuators embedded in distributed code, one can adaptively control software behavior either interactively or via fuzzy logic decision procedures. Both decision procedures and interactive systems can query managers with sensor and actuator attributes and retrieve global points to all sensors and actuators that match the specified attributes. *Autopilot's* Java-based Autodriver desktop interface, shown in Figure 2, allows users to attach dynamically to sensors and actuators, display time-varying sensor data, and change actuator values.

4 Project Accomplishments and Impact

The *Autopilot* research and resulting toolkit are recognized as major components of the nascent, multi-agency computational grid program. In particular, the *Autopilot* toolkit is a targeted technology of the National Computational Science Alliance's Partnership for an Advanced Computational Infrastructure (PACI) focus on wide-area computation and distributed collaboration. The *Autopilot* software targets optimization of computation

behavior when (a) executing on nation-scale distributed resources via the Globus toolkit and (b) using high-performance I/O systems on PC clusters.

Autopilot is also a major component of the DoE nuclear weapons stockpile stewardship program, via the Accelerated Strategic Computing initiative (ASCI). In particular, *Autopilot* technology is a part of DoE high-performance computing software research and development efforts. Similarly, our PPFS II adaptive I/O library, based on *Autopilot*, targets both analysis of I/O patterns in ASCI codes and dynamic optimization of these I/O patterns. Working with LLNL, LANL, and SNL, we are instrumenting laboratory codes and ASCI I/O libraries and developing adaptive I/O policies that support patterns found in these codes and libraries. In addition, lessons from PPFS II and *Autopilot* are helping drive creation of data and visualization corridors for the ASCI program. These corridors will support distributed analysis and visualization of petabyte data sets.

Autopilot is the basis for wide-area network tuning and optimization with the Globus toolkit, itself the base for the multiagency (NSF, NASA, DoE, and DARPA) computational grid. As part of the DoE NGI effort, we will be further integrating *Autopilot* and Globus for adaptive network control.

We also are working closely with contractors for the DoD High-Performance Computing Modernization Program (HPCCMP)² who are deploying our performance tools at DoD Major Shared Resource Centers (e.g. at CEWES) and using them to help DoD application Developers optimize their codes.

5 Software Availability

During its lifetime, the project released two major versions of the Autopilot software toolkit. Information on how to obtain the current version of the Autopilot software can be obtained via the World Wide Web at:

<http://www-pablo.cs.uiuc.edu/Projects/Autopilot/AutopilotOverview.htm>

² See <http://www.hpcmo.hpc.mil> for details.