Using Agents to Exploit Heterogeneous Parallelism on High Performance Computers

Mark H. Linderman
Air Force Research Laboratory/IFTC
26 Electronic Pkwy.
Rome, NY 13441
(315) 330-2164
(315) 330-2953 (FAX)
lindermanm@rl.af.mil

While homogeneous parallelism has traditionally been exploited on scaleable high performance computers (HPCs) for applications such as signal processing, parallelism in requiring data dependent processing applications is often difficult to predict and exploit with traditional methods. This paper describes the adaptation of 'agent-based' systems that have been investigated in other domains such as the Internet.

An agent is a autonomous process that adapts to its environment to accomplish a specific task. An agent first discovers the location of needed information, and then either sends the data to a central location or spawns processes to process the information in place. A process is spawned by encapsulating its executable code and state information (including how and where to send the results) into a package sent to the host compute node. The process is then scheduled and executed according to its priority.

Agents are used in distributed control applications to coordinate resource utilization for optimal performance, typically in a bandwidth-limited environment. The research presented here uses agents in a bandwidth-rich environment to dynamically allocate computational resources to several interacting tasks to maximize system performance. The flexibility of agent-based high performance computing can exploit the natural ebb and flow of computational requirements over time and space to maximize system performance in dynamically evolving environments.

C2I applications tend to be control flow oriented rather than data flow oriented. A query of an indexed database, for example, requires traversal of the index data structures involving many comparisons and conditional statements. Overall, the speed of the operation is determined by several factors including: whether the relevant portions of the index reside in core, the size of the database, how efficiently it is keyed for this query, and the particular records being sought. The cumulative effect of these factors is unpredictable query response time. A C2I system handling many simultaneous queries on a multiprocessor system has unpredictable load balancing dynamics. The semi-autonomous nature of agents allows them to adapt to constantly changing load conditions which improves load balancing and system performance.

The implementation of an agent system should be tailored to its problem domain and the architecture on which it is implemented. In many ways, the high performance computing agent systems are similar to the prevailing 'Internet' or Wide-Area Network (WAN) based systems: control of agent propagation and replication and coordination of agents. However, there are

important differences as well: communication bandwidth, security models and computational requirements.

The most obvious difference between an HPC and a Internet/WAN system is communication bandwidth. HPCs exhibit bisection bandwidths on the order of several gigabytes per second, in sharp contrast to the Internet with effective communication bandwidth of 1 KB/s to 1 MB/s. This extra bandwidth allows agents within the HPC to interact more closely and move about more freely. For example, a multi-part query may simultaneously launch several agents to seek out a preliminary data set from data available on the nodes of the HPC, and several other agents to receive the distributed preliminary dataset for further processing. Each of the agents may move about the nodes of the HPC in search of data and computational resources.

This example highlights one difference between traditional HPC applications and an agent-based system within an HPC. Traditional HPC applications typically exhibit communication patterns that are known when the application is spawned. Consequently, MPI v1.0, the de facto standard message passing application programming interface (API) for supercomputing supports only static communication patterns. Unfortunately, agents roam around the system in ways that are hard to predict, and therefore the communication infrastructure must be flexible enough to allow communication fabrics to evolve over time.

The second difference between the HPC and the Internet involves security and 'trust.' Agents operating over the Internet are typically untrusted, and therefore are not permitted access to storage devices (except in very carefully controlled situations), or to manipulate address pointers directly. Assumptions of trust are very different in a HPC environment. Users are authenticated when they log into the machine, and the programs that they execute are assumed benign. Furthermore, the authenticated user identity is associated with each agent/program, and agents may be granted extensive access to system resources (such as the file system) on a user by user basis, and the agents may be granted the privileges to which the user is entitled. In particular, if a user has many agents interacting, the agents may jointly manipulate data created by the user or the agents themselves.

The third way that an HPC-based architecture may differ from an Internet-based one is the manner in which resources are allocated to tasks. Because an HPC-based agent expects to interact directly with other agents and, it may inquire about the resources, agents and data resident on the node. The node and the agent may then jointly decide whether the agent should be allowed to run on that node and at what priority or whether it should go elsewhere. While there may be nothing precluding Internet-based agents from implementing a similar scheme, the heterogeneity of Internet resources may make it very difficult.

Several of the previously mentioned themes were incorporated into the Fusion Processor of the Discriminating Interceptor Technology Program (DITP). The Air Force Research Laboratory (AFRL) is developing this system for the Ballistic Missile Defense Organization (BMDO). The fusion tracker initiates, maintains, and discards detection tracks over time. Because it is impossible to predict the lifetime of a track, new work is dynamically allocated to maintain acceptable system throughput. A new detection is broadcast to the distributed track processes and the detection is associated with the track that best matches it. When a new track is

initialized, it is assigned to a node chosen to maintain an acceptable load balance with sufficient memory and compute resources to handle the track process.

When a track is migrated to a node, the data and code associated with that object are migrated with it and dynamically linked as a user process by the receiving node. A track manager catalogs the location (i.e. forwarding address) of the track process so that any process wanting to communicate can find it

While this work is a step in the direction of agents on a HPC, there are several aspects of an agent-based system that have not yet been addressed. First, the track processes are not semi-autonomous; they move and adapt only when directed by a centralized server. The 'authoritarian' control of the server is counter to the 'free-market' philosophy of agent-based systems. Second, an agent on a node does not reference any data belonging to other processes or interact with any other processes. Consequently, processes cannot be allocated to nodes based upon locality of related data. This lowers the efficiency of the system by creating more message traffic than necessary.

In the future, we intend to adapt agents to query dynamic databases resident upon an HPC. To derive maximal performance, the agents will be assigned prioritized tasks that may be comprised of subtasks. Because the resources to complete a task depend upon the type and quantity of data stored at each node, each task will determine the approximate computational requirements to accomplish that task on that node. The tasks will then be prioritized and spawned upon the nodes either sequentially or concurrently. The order and priority of a node task assignment will be influenced by several factors. For queries, these factors may include whether the query is existential or universal in nature. For database updates, the timeliness of the data is a factor in the priority assigned to the task.

Agent technology will dynamically exploit the natural time-varying resource requirements of C2I applications. It can be used to prioritize and balance the computational load to meet quality of service requirements. The introduction of agents to the HPC community will allow the tremendous computational capability of these machines to be harnessed for C2I applications and grows HPCs into new 'information-centric' application domains.