Appears in International Journal of Parallel Programming, Volume 20, Number 3, June 1991.

OREGAMI: Tools for Mapping Parallel Computations to Parallel Architectures [*]

Virginia M. Lo[†] Sanjay Rajopadhye[‡]
Samik Gupta, David Keldsen, Moataz A. Mohamed
Bill Nitzberg , Jan Arne Telle, and Xiaoxiong Zhong
Dept. of Computer and Information Science
University of Oregon
Eugene, Oregon 97403-1202
503-686-4408
lo@cs.uoregon.edu

Keywords: mapping, routing, embedding, task assignment,
regular parallel computations, parallel programming environments

Abstract

The OREGAMI project involves the design, implementation, and testing of algorithms for mapping parallel computations to message-passing parallel architectures. OREGAMI addresses the mapping problem by exploiting regularity and by allowing the user to guide and evaluate mapping decisions made by OREGAMI's efficient combinatorial mapping algorithms. OREGAMI's approach to mapping is based on a new graph theoretic model of parallel computation called the Temporal Communication Graph. The OREGAMI software tools includes three components: (1) LaRCS is a graph description language which allows the user to describe regularity in the communication topology as well as the temporal communication behavior (the pattern of message-passing over time). (2) MAPPER is our library of mapping algorithms which utilize information provided by LaRCS to perform contraction, embedding, and routing. (3) METRICS is an interactive graphics tool for display and analysis of mappings. This paper gives an overview of the OREGAMI project, the software tools, and OREGAMI's mapping algorithms.

[*]This research is sponsored by the NSF grant MIP91-08528 and the Oregon Advanced Computing Institute.
[†]Partially supported by NSF grant CCR-8808532
[‡]Partially supported by NSF grant MIP-8802454

1 Introduction

The *mapping problem* in message-passing parallel processors involves the assignment of tasks in a parallel computation to processors and the routing of inter-task messages along the links of the interconnection network. Most commercial parallel processing systems today rely on manual task assignment by the programmer and message routing that does not utilize information about the communication patterns of the computation. The goal of our research is *automatic* and *guided* mapping of parallel computations to parallel architectures in order to achieve portability and to improve performance.

The OREGAMI[1] project involves the design, implementation, and testing of mapping algorithms. OREGAMI's approach to mapping is based on (1) its use of a new graph theoretic model of parallel computation which we call the **T**emporal **C**ommunication **G**raph, and (2) its exploitation of regularity found in the structure of parallel computations and of the target architecture.

We are concerned with the mapping of parallel computations which are designed by the programmer as a collection of communicating parallel processes. We note that many practical parallel computations are characterized by regular communication patterns. This regularity occurs both in the topological communication structure of the computation (which tasks send messages to whom) and in the temporal communication behavior exhibited by the computation (the patterns of message-passing phases over time). Furthermore, the programmer has explicit knowledge of this regularity because it forms the basis of the logical design of her/his computation. While the general mapping problem in its many formulations is known to be NP-hard, it has been shown that restriction of the problem to regular structures can yield optimal solutions or good suboptimal heuristics for mapping.

OREGAMI addresses the mapping problem by exploiting regularity whenever possible, and by allowing the user to guide and evaluate mapping decisions made by OREGAMI's mapping algorithms. Our mapping system balances the user's knowledge and intuition with the computational power of efficient combinatorial mapping algorithms.

The contributions of the OREGAMI project include:

- **The TCG model of parallel computation and the LaRCS graph description language.** The TCG model is a new graph theoretic model of parallel computation designed for the purpose of mapping. The TCG can be seen as an augmented version of

[1]For University of **OREG**on's techniques for elegant symmetric contractions which resemble the art of ori**GAMI** paper folding.

Lamport's process-time graphs. The TCG integrates the two dominant models currently in use in the areas of mapping and scheduling: the static task graph and the DAG. LaRCS (**L**anguage for **R**egular **C**ommunication **S**tructures) is a graph description language which provides the user with the ability to describe the parallel computation's TCG in a natural and compact representation. LaRCS provides the capability to identify logically synchronous *phases* of communication, and to describe the temporal behavior of a parallel computation in terms of these phases in a notation called *phase expressions.*

- **Mapping algorithms which exploit regularity to yield high performance mappings.** Our mapping algorithms utilize the information provided by LaRCS to achieve mappings that are an improvement over uninformed mapping in two ways: the mapping algorithms themselves are efficient and the resultant mappings are optimal or near optimal based on a variety of standard performance metrics.

- **The OREGAMI software tools.** The OREGAMI tools include the LaRCS compiler; MAPPER, a library of mapping algorithms; and METRICS an interactive graphics tool for display and analysis of OREGAMI mappings.

  OREGAMI is designed for use as a front-end mapping tool in conjunction with parallel programming languages that support explicit message-passing, such as OCCAM, C*, Dino [33], Par [13], and C and Fortran with communication extensions. The underlying architecture is assumed to consist of homogeneous processors connected by some regular network topology, with current focus on the hypercube, mesh, and deBruijn topologies. Routing technologies supported by OREGAMI include store-and-forward, virtual cut-through, and wormhole routing. Systems such as the Intel iWarp, Intel iPSC machines, NCUBE hypercubes, and INMOS Transputer are candidates for use with OREGAMI. Note, however, that OREGAMI is a front-end mapping interface and generates symbolic mapping directives only. Development of back-end software to transform OREGAMI mapping directives to architecture dependent code is beyond the current scope of our work.

This paper provides an overview of the OREGAMI project and software tools. Section 2 discusses the formal foundations underlying OREGAMI's approach to mapping and related research. Section 3 describes the components of OREGAMI: LaRCS, MAPPER, and METRICS, and traces its operation with an illustrative example. Section 4 briefly illustrates how the individual mapping algorithms that we have developed for OREGAMI exploit regularity. Section 5 discusses areas of on-going and future work on this project.

2 Formal Foundations and Related Research

2.1 Formal Foundations: the Temporal Communication Graph

The foundations for the OREGAMI system lie in a new graph theoretic model of parallel computation we have developed called the **T**emporal **C**ommunication **G**raph and its ability to capture regularity for purposes of mapping. Both the TCG and its forerunner, the classic static task graph of Stone [41], are designed for systems in which the programmer designs his or her program as a set of parallel processes that communicate through explicit message passing. The identity of all of the processes is known at compile time, and all processes exist throughout the lifetime of the parallel computation.

The TCG was designed to enrich the static task graph model and to provide a means for describing regularity in the structure of the parallel computation for the purpose of mapping. Specifically,

- **The TCG integrates three important graph theoretic models of parallel computation: Lamport's process-time graphs, the static task graph, and the DAG.** The TCG combines the two predominant models currently in use in the areas of mapping, task assignment, partitioning, and scheduling: the static task graph and the precedence-constrainted DAG. The integration of these two models enables a wide spectrum of algorithms to be used for mapping and scheduling which could not otherwise be invoked because of incompatibilities in the underlying graph theoretic models. In addition, the compatibility of the TCG with Lamport's process-time graphs makes it useful as a unified abstraction in parallel programming environments for program development, debugging, and performance evaluation as well as for mapping and scheduling.
- **The ability of the TCG and the LaRCS graph description language to describe** *regularity* **facilitates the development and use of specialized mapping algorithms which exploit regularity to yield improved performance.** The TCG and LaRCS are capable of representing regularity both in the communication topology and in the temporal patterns of message passing over time.

2.1.1 Informal Description of the TCG

In this section, we give an intuitive description of the TCG. A formal definition of the TCG is given in [24] which also defines the formal semantics of LaRCS in terms of the TCG. Consider each individual process comprising the parallel computation. The activity of a given process

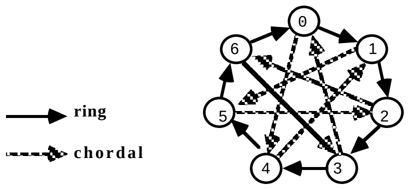
p_i can be seen as a sequence of atomic events, where each event is either a computation event or a communication event (sending a message/receiving a message). The TCG is a DAG in which each atomic event (compute, send, or receive) is represented as a node. The sequence of atomic events on p_i is represented as a linear chain of nodes, with directed edges indicating the precedence relationship between the events. A message-passing event from process p_i to process p_j is represented with a directed edge from the send-event node on p_i to the corresponding receive-event node on p_j.

The TCG can thus be seen as an unrolling of the static task graph over time. Conversely, the projection of the TCG along the time axis yields the static task graph model. Weights associated with the nodes and edges can be used to represent computation and communication costs, respectively. Note that the TCG also can be viewed as a graph theoretic representation of Lamport's process-time diagrams [21] augmented with weights and colors. The coloring of the Lamport process-time graph is described in [24] and involves the identification of logically synchronous communication and computation phases as described in the next section.

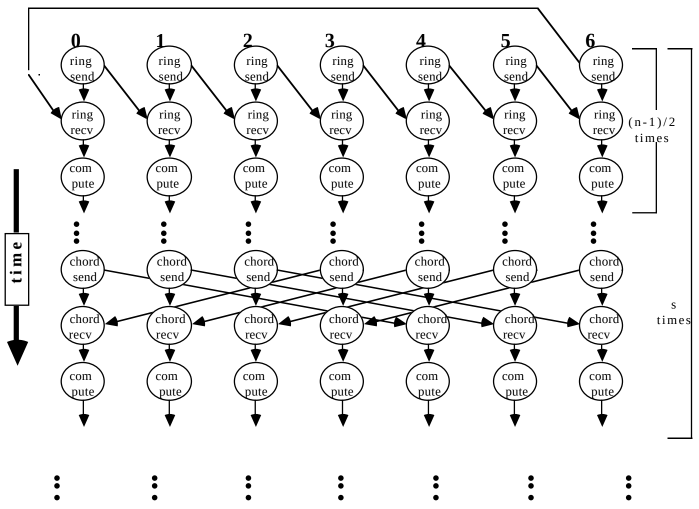
Figure 1 shows the TCG for a parallel algorithm for the *n-body* problem which was designed for the Cosmic Cube [37]. This algorithm will be used as an example later in the paper to illustrate the use of the OREGAMI mapping tools.

As discussed earlier, the TCG can be seen as a hybrid of the two predominant models of parallel computation: the static task graph model of Stone [41], and the precedence-constrained (DAG) model [30] used in multiprocessor scheduling and in the parallelization of sequential code. Task assignment and scheduling research utilizing these two models has more or less followed disjoint paths over the past two decades, in that techniques and algorithms developed for one model have not been applicable to the other. The TCG model is compatible with both of these existing models. Thus algorithms for static task assignment such as [8] [6] [23], [35] and scheduling algorithms for precedence-constrained graphs such as [30] [9] can be applied to the TCG model. The contribution of the TCG is that it augments these two models with the ability to explicitly capture regularity, allowing the development of specialized mapping and scheduling algorithms to exploit this regularity.

The TCG is also capable of modeling computations of arbitrary granularity, characterized by irregular and asynchronous communication. We note that the TCG does not model non-deterministic computations and dynamically spawned tasks. More information about the TCG model and its use in parallel programming environments can be found in [24].



(a) 7-body static graph



(b) TCG of 7-body

Figure 1: Temporal communication graph for the *n-body algorithm*

The *n-body problem* requires determining the equilibrium of n bodies in space (where n is odd) under the action of a gravitational or electrostatic, etc. field. This is done iteratively by computing the net force exerted on each body by the others, given their "current" position, updating its location based on this force, and repeating this until the forces are as close to zero as desired. The parallel algorithm presented by Seitz uses Newton's third law of motion to avoid duplication of effort in the force computation. It consists of n identical tasks, each one responsible for one body. The tasks are arranged in a ring and pass information about their accumulated forces to its neighbor around the ring. After $(n-1)/2$ steps, each task will have received information from half of its predecessors around the ring. Each task then acquires information about the remaining bodies by receiving a message from its chordal neighbor halfway around the ring. This is repeated to the desired degree of accuracy.

2.1.2 Describing Regularity in the TCG

We note that many practical parallel algorithms involve one or more *phases* of communication. These phases are often characterized by regularity in the communication topology, i.e., the static task graph is a known graph structure such as a mesh, a tree, etc. In addition, these algorithms also exhibit regularity in the temporal communication behavior (the patterns of message passing phases that are active over time).

A *compute phase* corresponds to a set of nodes in the TCG (compute events) that are involved in *logically synchronous* computation. A *communication phase* corresponds to a set of edges (sender/receiver pairs) in the TCG that are involved in *logically synchronous* communication. By *logically synchronous* we mean that at run time the activities occur simultaneously from the viewpoint of the programmer, i.e. from the logical structural design of the algorithm.[2]

These phases are identified using a node labeling scheme for each process. Each communication phase can then be described by a *communication function* whose domain and range are process node labels. For example, a ring of communicating processes can be described as `ring: forall i in 0..n-1 ⇒ i+1 mod n` where we assume the processes are labeled sequentially from `0` to `n-1`. In LaRCS, a notation called *phase expressions* is used to describe the temporal behavior of the parallel computation in terms of its compute phases and communication phases. In addition, LaRCS allows the user to describe families of regular graphs in a parameterized notation whose size is independent of the size of the task graph. The LaRCS description of the *n-body algorithm* and further details about LaRCS are described in the next section.

2.2 Related Research

As an integrated set of tools for mapping, OREGAMI is similar in many respects to Francine Berman's Prep-P System [4] [6]. Both Prep-P and OREGAMI provide a graph description language for describing the communication structure of the parallel computation. Prep-P's GDL language is based on the static task graph model of parallel computation and is embedded within the system's parallel programming language XX. LaRCS is based on our TCG model, which augments the static task graph with temporal information. The LaRCS specification is independent of any specific parallel programming language and is coded separately by the programmer. Prep-P is a fully integrated system: Prep-P mappings are targeted for the CHiP reconfigurable parallel architecture [38] and Prep-P currently generates code that runs on the CHiP simulator known

[2]In reality, when the program executes, the timing of logically synchronous activities may not be synchronous with respect to real time, due to effects such as the hardware characteristics of the execution environment and the multiplexing of processes on the processors.

as Poker. OREGAMI also is related to the TMAP mapping tool, a component of the TIPS transputer-based interactive parallelizing system [44]. TMAP is an adaptation of Prep-P for the transputer architecture. As discussed in the introduction, OREGAMI is currently a front-end mapping tool and only generates symbolic mapping directives.

OREGAMI is also related to a number of mapping and scheduling systems that utilize the classic DAG model of parallel computation such as CODE/ROPE [10], TaskGrapher [14], Polychronopoulos [30] and Sakar [36]. These systems differ from OREGAMI, Prep-P, and TMAP because they are designed for the purpose of parallelizing sequential code.

The OREGAMI mapping algorithms take the approach of many researchers who have attacked the mapping problem by exploiting the regularity found in the computation graph and/or the interconnection network. A large body of theoretical work on graph embeddings has yielded 1:1 mappings tailored for specific regular graphs; some of the more recent results are given in [32]. Many of these algorithms have or will be included in the OREGAMI library. Edge grammars [5] use formal language techniques to address the contraction of families of task graphs. Stone and Bokhari [41], [8] use a variety of graph theoretic algorithms to address task assignment for structures including trees, chains, and arbitrary task graphs. Work in the area of systolic arrays has yielded elegant mapping techniques for computations whose data dependencies can be expressed as affine recurrences [31] [12]. Related work in the area of application-dependent routing includes [19] and [7]. Our mapping algorithms build on these foundations and utilize techniques from group theory, graph theory, coding theory, and linear algebra. Other approaches to the mapping problem include search algorithms, linear programming, and clustering algorithms. These latter techniques typically do not exploit the regularity of the task graph.

The LaRCS language bears similarities to a number of graph description languages or configuration languages which have been developed for a variety of purposes in the area of parallel and distributed computing. These include formal approaches such as edge grammars [5] and graph grammars [20], [2], as well as more practical languages such as GDL [4], CONIC [28], GARP [20], and ParaGraph [3]. Of these, LaRCS, edge grammars, and GDL were designed specifically to address the mapping problem. LaRCS is unique in its ability to describe the temporal behavior of parallel computations. A primitive form of phase expressions was introduced by [29].

¿From the viewpoint of support environments for parallel programming, OREGAMI belongs to the family of systems which take a process-oriented view of parallel computation based on explicit message-passing. Examples of such systems include Prep-P [4, 6], Poker [38, 40], ORCA [39, 16], the Parallel Programming Environments Project [3], and TIPS [44]. OREGAMI and Prep-P focus on the mapping problem. The other projects address the broader issues of program design and development.