Performance Indices for Parallel Marker-Propagation [†]

R. F. DeMara and D. I. Moldovan
Dept. of Elec. Engr.–Systems, University of Southern California
Los Angeles, California 90089-1115

Abstract

Marker-propagation is an intrinsically parallel reasoning technique which has been used in many AI applications. Recent research has concentrated on how to implement marker-propagation efficiently using existing and special-purpose parallel computer architectures [1] [2] [3]. However, few measures currently exist to quantify the computational work performed in these systems. Existing metrics such as Logical Inferences per Second (LIPS) obscure the computational features of this massively parallel approach. We propose a set of indices for measuring processing throughput and characterizing the associated knowledge bases.

Propagation Mechanisms

Our objective was to develop representative processing indices which could be related to features of the application program. These metrics should be similar in concept to those which already exist for numeric-processing applications, such as the number of 32-bit floating-point operations (FLOPs) performed. In numeric processing the basic floating point operation is a representative metric because it forms a common basis between the algorithm and machine. However, no equivalent measures have been available for marker-propagation.

We have approached this problem by defining metrics capable of quantifying the fundamental operations of "useful work" under the marker-propagation paradigm. We contend that each marker propagation has two primary effects:

1. *Conditioning:* activation or deactivation of the status bit in a node or the execution of an arithmetic operation upon node or link registers,
2. *Spreading:* propagation of the marker to other selected nodes specified by some propagation rule.

During propagation, spreading can be considered transient work while node conditioning produces steady-state effects. This is because spreading consists of search-and-dispersion operations which consume computational resources, but have no lasting impact on the knowledge base. On the other hand, conditioning operations modify the state of the knowledge base during execution. Our set of indices for marker-propagation are listed in Table 1.

[†] This research has been funded by National Science Foundation Grants MIP-89/02426 and MIP-90/09109

Propagation Indices		Significance
$\mathcal{A}$	marker activation	changes to knowledge base
$\mathcal{M}_{inj}$	marker injection	new markers introduced
$\mathcal{S}(t)$	scattering function	time behavior behavior of routing and replication
$\mathcal{I}$	marker inspection	searching work performed
$\mathcal{D}$	marker dispersion	spreading work performed

Table 1: Indices for Marker-Propagation.

Marker Activation

The *activation*, $\mathcal{A}$, of an injected marker is a measure of the effect it has on the knowledge base. It is a computation-oriented metric. This quantity is the sum of the activation status bits flipped at a node (these bits that indicate membership in a hypothesis, i.e. the nodes become marked by them) and the number of FLOPs induced (arithmetic computations to compute marker value). Thus given a marker propagation m_i we measure:

$$\mathcal{A}_{m_i} = \sum_{\substack{\text{prop cycle} \\ m_i}} (\text{bit inversions}) + \sum_{\substack{\text{prop cycle} \\ m_i}} \left(\begin{matrix} \text{FLOPs} \\ \text{performed} \end{matrix} \right) \tag{1}$$

Thus the number of activations per second is a microscopic indicator of the amount of computational work performed, in the absence of communication expenses.

Marker Scattering Function

The marker *scattering function*, $\mathcal{S}^t_{m_i}$, denotes the time behavior of two figures-of-merit for work done by spreading. It reflects the communication work performed. The first component, called the *inspection*, is defined as:

$$\mathcal{I}_{m_i} = \sum_{\substack{\text{prop cycle} \\ m_i}} \sum_{\substack{\text{nodes} \\ \text{encountered}}} \left(\begin{matrix} \text{relations} \\ \text{searched} \end{matrix} \right) \tag{2}$$

It is the amount of work performed by searching the relations in accordance with the propagation rule. The second component of the scattering function is the *dispersion*. The dispersion measures the amount of new

propagations that are recursively created:

$$\mathcal{D}_{m_1} = \sum_{\substack{prop\ cycle \\ m_1}} \left(\overset{new}{activations} \right) \quad (3)$$

This is precisely the amount of successful inspections:

$$= \sum_{\substack{prop\ cycle \\ m_1}} \sum_{\substack{nodes \\ encountered}} (\mathcal{I}_{relation} = \mathcal{I}_{prop\ rule}) \quad (4)$$

Thus the scattering function for the j^{th} propagation is:

$$\mathcal{S}^j_{m_1} = \mathcal{I}^j_{m_1} + \mathcal{D}^j_{m_1} \quad (5)$$

The motivation for separating the components of the spreading tasks is to isolate the effects of particular algorithm features. The distinction may be significant in the context of the architecture used. For example, suppose we wanted to assess the suitability of Content Addressable Memory (CAM) in a marker-propagation machine. These metrics could be tabulated for two application algorithms φ_1 and φ_2 as part of the design study. The algorithm $\varphi_1 = \langle\ large\ \mathcal{I}_{m_1}\ ,\ small\ \mathcal{D}_{m_1}\ \rangle$ would benefit from a CAM for fast searching during the inspection process. On the other hand, $\varphi_2 = \langle\ small\ \mathcal{I}_{m_1}\ ,\ large\ \mathcal{D}_{m_1}\ \rangle$ would benefit from tighter coupling between nodes, but derive little benefit from a CAM.

Lazy Markers

Under the marker-passing model, it is not possible to know a-priori how long a marker-propagation instruction will take. A major problem this can cause is low resource utilization while waiting for the final markers to complete propagation. We have called this the *lazy marker effect*. We refer to the responsible marker as a *lazy marker*. To quantify this effect, it is necessary to specify precisely what it means to wait a "long" time. This may be readily defined in terms of the scattering function $\mathcal{S}(t)$. Here

$$\mathcal{S}(t)\ \forall\ t_{injection} \leq t \leq t_{end\ propagation} \quad (6)$$

indicates the markers propagating through the network during the propagation period. Lazy effects occur at the tail-end of this distribution. We know that $\mathcal{S}(t = t_{injection}) = 1$ and $\mathcal{S}(t = t_{end}) = 0$. However, $\mathcal{S}(t)$ is not a strictly bitonic function. Thus we define lazy markers in terms of the length of the propagation cycle spent waiting for the last 1% of the markers to finish.

$$t_{lazy} \triangleq t_{99-100}\ such\ that\ \mathcal{S}(t) \leq 1\%\ of\ \mathcal{S}_{max} \quad (7)$$

The t_{lazy} parameter must be considered relative to the overall propagation period T_{prop}. Consider the bursty nature of reasoning algorithms which generate ratios of $(T_{prop} - t_{lazy})/T_{prop} \approx 0.75$. This indicates for at least 25% of the time, the machine is operating at less than 1% of capacity. In this case, the price one has to pay to provide massively parallel support for marker-propagation is low processor utilization on average.

Network Indices		Significance
F_{out}	fan-out	outgoing relations per node
F_{in}	fan-in	incoming relations per node
$l_{reltype}$	path length	relations in a chain of links of type *reltype*
M	network size	concepts in hierarchy
$N_{reltype}$	composition	(number of *reltype* links) $\div M$
T	topology	$T \epsilon \{tree, mesh, random$ *replicated subgraph*$\}$

Table 2: Networked Representation Indices.

Networked Knowledge Bases

Marker-propagation algorithms typically represent knowledge using semantic networks. Semantic networks are directed graphs of concepts (graph nodes), their properties (graph links), and the hierarchical relationship (partial ordering) between them. Semantic networks are difficult to characterize because their interconnections are essentially random. To characterize them, we use the average and maximum values of the parameters in Table 2.

We have observed a common feature that we call the *replicated subgraph*. This refers to the existence of large forests of similar trees, with each tree representing an specific event sequence. This regularity of structure provides insight into likely hot spots during computation.

We feel three cases are especially representative for benchmarking marker-propagation performance:

1. vary M (e.g. 256, 1K, 4K, 16K nodes) with T fixed (tests the scale-up of a similar but larger knowledge base),
2. vary $l_{reltype}$ with F_{in}, F_{out} fixed (examines effect of changing the critical path), and
3. vary M and F_{out} with $l_{reltype}$ fixed (examines impact of network bushyness).

We are currently analyzing knowledge bases in terms of these parameters. First, we would like to obtain some idea of what typical ranges are for these indices. Second, this will allow construction of a small set of standardized, synthetic knowledge bases for benchmarking purposes.

References

[1] R. F. DeMara and D. I. Moldovan, "The SNAP-1 Parallel AI Prototype," in *Proceedings of Eighteenth Annual International Symposium on Computer Architecture*, 1991.

[2] M. Evett, J. Hendler and L. Spector, "PARKA: Parallel Knowledge Representation on the Connection Machine," Technical Report UMIACS-TR-90-22, University of Maryland, 1990.

[3] T. Higuchi, *et al*, "The Prototype of a Semantic Network Machine IXM". *Proceedings of 1989 International Conference on Parallel Processing*.