Higher-level parallel programming models express parallelism in an implicit way. Instead of imposing programmers to create multiple tasks that can run concurrently and handle their communications and synchronizations explicitly, these models allow programs to be written without assumptions of artificial sequentiality. The programs are naturally parallel. Examples of such kind of models include the Chemical Reaction Models (CRMs) (Banatre & Le Metayer, 1990, 1993), Linda (Carriero & Gelernter, 1989), and Unity (Chandy & Misra, 1988; Misra, 1989). These models are created to address higher level programming issues such as formal program specification, program synthesis, program derivation and verification, and software architecture. Efficient implementation
of these models has limited success and therefore obscures its direct applications in software design (Creveui, 1991; Gladitz, 1996). Despite this limitation, efforts have been made in both academic and industrial settings to avail these models in real-world programming. For example, Unity has been used in industrial software design and found successful; execution efficiency of Linda has been affirmed by experiments and it is implemented by IBM Tuple Space. Recent discussions of these models in multi-agent system design have also been found in literature (Cabri, 2000). In the following discussion, we focus on the Chemical Reaction Models and its applications.

The Chemical Reaction Models describe computation as “chemical reactions”. Data (the “solution”) are represented as a multiset. A set of “reaction” rules is given to combine elements in the multiset and produce new elements. Reactions take place until the solution becomes inert, namely there are no more elements can be combined. The results of computation are represented as the inert multiset. Gamma is a kernel language in which programs are described in terms of multiset transformations. In Gamma programming paradigm, programmers can concentrate on the logic of problem solving based on an abstract machine and are free from considering any particular execution environment. It has seeded follow-up elaborations, such as Chemical Abstract Machine (Cham) (Berry & Boudol, 1992), higher-order Gamma (Le Metayer, 1994; Cohen & Muylaert-Filho, 1996), and Structured Gamma (Fradet & Le Metayer, 1998). While the original Gamma language is a first-order language, higher order extensions have been proposed to enhance the expressiveness of the language. These include higher-order Gamma, hmm-calculus, and others. The recent formalisms, $\gamma$-Calculi, of Gamma languages combine reaction rules and the multisets of data and treat reactions as first-class citizens (Banâtre, Fradet, & Radenac, 2004, 2005a, 2005b). Among $\gamma$-Calculi, $\gamma_0$-Calculus is a minimal basis for the chemical paradigm; $\gamma_c$-Calculus extends $\gamma_0$-Calculus by adding a condition term into $\gamma$-abstractions; and $\gamma_n$-Calculus extends $\gamma_0$-Calculus by allowing abstractions to atomically capture multiple elements. Finally, $\gamma_{cn}$-Calculus combines both $\gamma_c$-Calculus and $\gamma_n$-Calculus. For notational simplicity, we use $\gamma$-Calculus to mean $\gamma_{cn}$-Calculus from this point on.

The paper will be organized as follows. In the second section, we give a brief introduction to $\gamma$-Calculus. In the third and the fourth section, we discuss the method for implementing $\gamma$-Calculus in IBM Tuple space and in OpenCL, respectively. Experimental results are presented thereafter. We conclude in the last section.

$\gamma$-CALCULUS

The basic term of a Gamma program is molecules (or $\gamma$-expressions), which can be simple data or programs ($\gamma$-abstractions). The execution of the Gamma program can be seen as the evolution of a solution of molecules, which react until the solution becomes inert. Molecules are recursively defined as constants, $\gamma$-abstractions, multisets or solution of molecules. The following is their syntax:

$$M::= 0 | 1 | \ldots | \text{‘a’} | \text{‘b’} | \ldots ; \text{constants}$$
$$| \gamma P[C].M ; \gamma\text{-abstraction}$$
$$| M_1, M_2 ; \text{multiset}$$
$$| <M> ; \text{solution}$$

The multiset constructor “,” is associative and commutative (AC rule). Solutions encapsulate molecules. Molecules can move within solutions but not across solutions. $\gamma$-abstractions are elements of multisets, just like other elements. They can be applied to other elements of the same solution if a match to pattern $P$ is found and condition $C$ evaluates to true and therefore facilitate the chemical reaction. The pattern has the following syntax:
Related Content

An Exploratory Study on Policy Transfer for SIM Card Registration in Malawi
[www.igi-global.com/article/an-exploratory-study-on-policy-transfer-for-sim-card-registration-in-malawi/125593?camid=4v1a](www.igi-global.com/article/an-exploratory-study-on-policy-transfer-for-sim-card-registration-in-malawi/125593?camid=4v1a)

Competition on the Web Market Both Against Other Web Based Market Companies and Against Their Real Market Analogues
[www.igi-global.com/article/competition-on-the-web-market-both-against-other-web-based-market-companies-and-against-their-real-market-analogues/114971?camid=4v1a](www.igi-global.com/article/competition-on-the-web-market-both-against-other-web-based-market-companies-and-against-their-real-market-analogues/114971?camid=4v1a)

Ndjebbana Talking Books: A Technological Transformation to Fit Kuynibidji Social Practice
[www.igi-global.com/chapter/ndjebbana-talking-books/23554?camid=4v1a](www.igi-global.com/chapter/ndjebbana-talking-books/23554?camid=4v1a)

Student Technology Projects in a Remote First Nations Village
[www.igi-global.com/chapter/student-technology-projects-remote-first/23546?camid=4v1a](www.igi-global.com/chapter/student-technology-projects-remote-first/23546?camid=4v1a)