Templet Parallel Computing System: Specification, Implementation, Applications

Sergei Vostokin
Professor, Information Systems and Technologies Department, Samara University

ITNT-2017, Samara, 25-27 April, 2017
The Temple system aims to reduce the parallel programming complexity to the level of sequential programming for ones who develop their own HPC applications.

Programming in Templet means:
• a view of parallel algorithm
  sequential algorithm + specification of parallelism
  \[\rightarrow\] parallel algorithm
• using standard language (C++), libraries (OpenMP, MPI), IDE, and the Templet lightweight tooling in application development

Research methods:
• actor model of execution (Carl Hewitt)
• temporal logic of actions (Leslie Lamport)
• algorithmic skeletons (Murray Cole)
• language-oriented programming (Martin Ward)
IT IS IMPORTANT TO UNDERSTAND HOW AN APPLICATION INTERACTS WITH ITS EXECUTION ENVIRONMENT

Parallel applications

- Poincaré map
- Continuous wavelet transform
- Finite difference method
- Business process modeling
- Others

Mathematical specification of computation

Run-time implementations

- Debugging
- Sequential execution
- Parallel execution in shared memory
- Parallel execution in distributed memory
- Emulated execution
MAPPING FROM A VARIABLE TO ACTOR OR MESSAGE

\[ F : \text{Var} \rightarrow \{\text{actor, message}\} \times \mathbb{N} \]

an actor

a message
\{a[1], a[2], \ldots, a[i], \ldots, b[1], b[2], \ldots, b[j], \ldots, m[1], m[2], \ldots, m[j], \ldots\}

\[
a[i] \rightarrow \neg a[i]
\]

\[
b[j] = i
\]

\[
m[j] \rightarrow \neg m[j]
\]
SEQUENCE OF STATES (ACTIONS) IN THE TRANSMISSION OF A MESSAGE BETWEEN TWO ACTORS

S(0) → A_2 \land A_3 → S(1) → A_1 \land A_4 → S(2)
A1: ACTOR STARTS THE EXECUTION

\[ A_1 \equiv \exists! j : \neg a[i] \land m[j] \land b[j] = i \land a'[i] \land \neg m'[j] \land b'[j] = i \]
\[ A_2 \equiv a[i] \land \neg a'[i] \]
\[ A_3 \equiv \exists i : a[i] \land b[j] = i \land \neg m[j] \land m'[j] \]
$A_4 \equiv m[j] \wedge \neg m'[j]$
ADDITIONAL DEFINITIONS

\[ I \equiv \exists i : a[i] \lor \exists j : m[j] \]

\[ f_1 \equiv a[i] \quad f_2 \equiv (m[j], b[j]) \]

\[ \square [A]_f \] means that the action \( A \lor (f = f') \) executes for every pairs of system states.

\[ WF_f (A) \] means that if the action \( A \) (that change variables \( f \)) is enabled long enough it will finally be executed.
\[ S \equiv I \land \Box \left[ A_1 \lor A_2 \right]_{f_1} \land \Box \left[ A_3 \lor A_4 \right]_{f_2} \land WF_{f_1}(A_2) \land WF_{f_2}(A_4) \]
The message handler procedure $\text{RECV}(i,j)$ is called with the action

$$
\text{recv}_{(\text{call})}(i, j) \equiv \neg a[i] \land m[j] \land b[j] = i \\
\land a'[i] \land \neg m'[j] \land b'[j] = i
$$

and returns with the action

$$
\text{recv}_{(\text{return})}(i) \equiv a[i] \land \neg a'[i]
$$

of the actor run-time system $S$. 
Testing for the accessibility of a message

\[
\text{access} (i, j) \equiv b[j] = i \land \neg m[j]
\]

Sending a message to an actor

\[
\text{send} (i, j) \equiv b'[j] = i \land m'[j]
\]
(A) the handler $RECV(I,\_)$ can access actor state variables $var$, if $F(var)=I$

(B) the handler $RECV(I,\_)$ can access message state variables $var$, if $F(var)=J$ and $access(I,J)=true$

(C) the handler $RECV(I,\_)$ can send a message with the call $send(\_,J)$, if $access(I,J)=true$
// engine
struct engine { std::vector<message*> ready; };
// actor objects
struct actor { void (*recv)(actor*, message*); };
// message objects
struct message { actor*a; bool sending; };

inline void send(engine*e, actor*a, message*m) {
    if (m->sending) return;
    m->sending = true;
    m->a = a;
    e->ready.push_back(m);
}
inline bool access(actor*a, message*m) {
    return m->a == a && !m->sending;
}
inline void run(engine*e) {
    size_t rsize;
    while (rsize = e->ready.size()){
        int n = rand() % rsize;
        auto it = e->ready.begin() + n;
        message*m = *it; e->ready.erase(it);
        m->sending = false;
        m->a->recv(m->a,m);
    }
}
For more details, please, see the poster section presentation:

S. Vostokin, E. Skoryupina
A Performance Analysis of Simple Runtime System for Actor Programming in C++
THANK YOU

http://templet.ssau.ru

https://Github.com/templet-language

E-mail: Sergei Vostokin <easts@mail.ru>