Computation Orchestration

Jayadev Misra

Department of Computer Science
University of Texas at Austin

Email: misra@cs.utexas.edu
web: http://www.cs.utexas.edu/users/psp

Lecture at SEFM 2004, Beijing.
September 28, 2004

Computation Orchestration

Given are basic computing elements. How to compose them?

- Computing elements are logic gates: $\wedge$, $\vee$, $\neg$

  Composition is a circuit.

- Computing elements are functions.

  Composition is through higher-order functions.

- Computing elements are processes.

  Composition is through CCS or CSP operators.

Orc

Computing elements are Sites, such as

- function: Compress MPEG file
- method of an object: LogOn procedure at a bank
- monitor procedure: read from a buffer
- web service: get a stock quote
- transaction: check account balance
- distributed transaction: move money from one bank to another

Lecture Material

Computation Orchestration: A Basis for Wide-Area Computing

http://www.cs.utexas.edu/users/psp/Wide-area.pdf

To appear as two chapters in the Proceedings of

The NATO International Summer School, Marktoberdorf, Germany.

Example: Airline

- Contact two airlines simultaneously for price quotes.
- Buy ticket from either airline if its quote is at most $300.
- Buy the cheapest ticket if both quotes are above $300.
- Buy any ticket if the other airline does not provide a timely quote.
- Notify client if neither airline provides a timely quote.

Example: workflow

- An office assistant contacts a potential visitor.
- The visitor responds, sends the date of her visit.
- The assistant books an airline ticket and contacts two hotels for reservation.
- After hearing from the airline and any of the hotels: he tells the visitor about the airline and the hotel.
- The visitor sends a confirmation which the assistant notes.

Example: workflow, contd.

After receiving the confirmation, the assistant

- confirms hotel and airline reservations.
- reserves a room for the lecture.
- announces the lecture by posting it at a web-site.
- requests a technician to check the equipment in the room.

Wide-area Computing

Acquire data from remote services.

Calculate with these data.

Invoke yet other remote services with the results.

Additionally

Invoke alternate services for failure tolerance.

Repeatedly poll a service.

Ask a service to notify the user when it acquires the appropriate data.

Download an application and invoke it locally.

Have a service call another service on behalf of the user.

The Nature of Distributed Applications

Three major components in distributed applications:

Persistent storage management

databases by the airline and the hotels.

Specification of sequential computational logic

does ticket price exceed $300?

Methods for orchestrating the computations

contact the visitor for a second time only after hearing from the airline and one of the hotels.

We look at only the third problem.

Orc

A new kind of assignment

$x :\in f$

where x is a variable and f is an Orc expression.

Evaluation of f yields zero or more values.

Assign the first value to x.

An Orc expression is

- Simple: Site (Function call, method, web service, transaction)
- Compound: $f \mid g$, $f \gg g$, f^*, $\{ f \;\textbf{where}\; x :\in g \}$

Simple Orc Expression

- M is a news service, d a date. Download the news page for d.

  $x :\in M(d)$

- Side-effect: Book ticket at airline A for a flight described by c.

  $x :\in A(c)$

The returned value is the price and the confirmation number.

Properties of Sites

- A site may not respond.

  Its response at different times (for the same input) may be different.

- A site call may change states (of external servers) tentatively or permanently.

  Tentative state changes are made permanent by explicit commitment.

Structure of response

- The response from a site has:
  value, which the programmer can manipulate, and
  pledge, which the programmer cannot manipulate.

- Pledge is used to commit this site call.
  Pledge is valid for some time period.
  Value is meaningful during then.

- By committing a valid pledge (during the given period), the programmer establishes some fact.

Nesting

- (Data Piping) Retrieve a news page for date d from M and email it to address a. Here, $Email$ is a site.

$$Email(a, M(d))$$

- (Higher-order site) Call discovery service D with parameter x to locate a site; call that site with parameter y.

$$Apply(D(x), y)$$

Simple Orc Expression: Sequencing

M, N, R are sites for 3 professors.

s is a set of possible meeting times.

$M(s)$ is a subset of s, the times when M can meet.

$M(N(R(s)))$ is the possible meeting times of all three professors.

Parallel, Strict evaluation

Arguments of a site call are evaluated in parallel.

A site is called only after all its arguments have been evaluated.

Fork-join parallelism

$A(c)$ and $B(c)$ return ticket prices from airlines A and B.

Min returns the minimum of its arguments.

$Min(A(c), B(c))$:

Compute $A(c)$ and $B(c)$ in parallel.

Call Min when both quotes are available.

Predefined sites

- $Fail$ never responds.
- $let(x, y, \cdots)$ returns a tuple of argument values as soon they are available. $let(\theta)$ is $skip$.
- $random$ returns a random number (in a specified range), instantaneously.
- fst returns the value of the first argument as soon all argument values are available.
- $timer(t)$, where t is a non-negative integer, returns a signal exactly after t time units.
- $timer(t, x)$ is $fst(x, timer(t))$; returns x after t time units.

Composing Expressions

- (Alternation) $f \mid g$: evaluate f and g in parallel; values of $f \mid g$ are those from f and from g.
- (Piping) $f \gg g$: Evaluate g for all values of f; values of $f \gg g$ are those from g.
- (Iteration) f^*: values from f after zero or more piping steps.

$$\begin{array}{rl} & f^* \\ = & \mathbf{1} \mid (f \gg f^*) \\ = & \mathbf{1} \mid (f \gg (\mathbf{1} \mid (f \gg (\mathbf{1} \mid f \gg \cdots)))) \end{array}$$

- (Definition) $\{\ f\ \mathbf{where}\ x{:}{\in}\ g\}$

Binding power

$|$ has the lowest binding power, then $\gg$ and $*$ has the highest binding power.

$$f^* \ | \ h \gg g \ \equiv \ (f^*) \ | \ (h \gg g)$$

Example of Orc expression:

$$G(q) \gg (\ \langle M(q) \mid R(\theta, q) \gg G(\theta) \rangle^* \gg S(\theta) \ )$$

Programming Notes:

The expression will be written in a more understandable way.

Default Parameter

- $M \gg N(x, \theta)$

- $(M \mid S) \gg (N(x, \theta) \mid R(\theta))$

- Start computation of f with value v for θ:
  $let(v) \gg f$.

- Start an iteration where $x_0 = v$ and $x_{i+1} = M(x_i)$.
  Values returned are $N(x_i)$, for $i \geq 0$.

$$let(v) \gg M(\theta)^* \gg N(\theta)$$

Alternation, Piping

- Assign the first value from $M(c)$ or $N(d)$ to z.

$$z :\in M(c) \mid N(d)$$

- assign to z the value from M if it arrives before t, 0 otherwise.

$$z :\in M \mid timer(t, 0)$$

- Interruption

$$f \mid Interrupt.get$$

- Make four requests to site M, in intervals of one time unit each.

$$M \mid timer(1) \gg M \mid timer(2) \gg M \mid timer(3) \gg M$$

Priority

Request M and N for values. Give priority to M.

- Allocate one extra time unit for M to respond.

$$z :\in M \mid timer(1) \gg N \quad \text{or} \quad z :\in M \mid N \gg timer(1, \theta)$$

- Accept the response from M if it arrives within one time unit, else accept the first response.

$$z :\in M \mid fst(N, timer(1))$$

Iteration

- Call M forever at unit time intervals until it returns a value.

  $$z{:}\in timer(1)^* \gg M$$

  which is

  $$z{:}\in M \mid timer(1) \gg (M \mid timer(1) \gg (M \mid \cdots))$$

- Same as above, but stop calling after 10 time units.

  $$z{:}\in timer(1)^* \gg M \mid timer(10)$$

Iteration; Contd.

- Site M returns stock price of company abc

  Site $C(x)$: returns x if $x < 20$; silent otherwise.

  $$M^* \gg C(\theta)$$

  either never returns a value (if abc never falls below 20)

  or returns a value lower than 20. Initially, $\theta \geq 20$.

- Variation: Poll M once every hour for 6-hours:

  $$timer(1)^* \gg \langle M \gg C(\theta) \rangle \ | \ timer(6)$$

Definition within Orc expression

- A machine is assembled from two parts, u and v.
- Two vendors for each part: $u1$ and $u2$ for u, and $v1$ and $v2$ for v.
- Solicit quotes from all vendors.
- Accept the first quote for each part.
- Compute the machine cost to be 20% above the sum of the part costs.

$cost{:}\in$ $\{$ $(u + v) \times 1.2$
where
$u{:}\in u1 \mid u2$
$v{:}\in v1 \mid v2$
$\}$

General Orc Statements

$$
\begin{array}{l}
z{:}\in \{\;\; f(\cdots x \cdots y \cdots) \\
\qquad \textbf{where} \\
\qquad\qquad x{:}\in\; g \\
\qquad\qquad y{:}\in\; h \\
\quad \}
\end{array}
$$

Example: M, N, R, S are sites.

$$
\begin{array}{ll}
z{:}\in \{\;\; (M(x) \mid N(y)) \gg M(y) & \gg \;\{\;\; M(y) \\
\qquad \textbf{where} & \qquad \textbf{where} \\
\qquad\qquad x{:}\in\; R(y) \mid N(y) & \qquad\qquad y{:}\in\; S \\
\qquad\qquad y{:}\in \{\;\; R \mid N(t) & \;\} \\
\qquad\qquad\qquad \textbf{where} \quad t{:}\in\; S & \\
\qquad\qquad\quad \} & \\
\quad \} &
\end{array}
$$

Fork-Join parallelism

$z{:}\in fst(\textit{true}, x)$ $\quad|\quad$ $fst(\textit{false}, x)$

where $\qquad\qquad$ **where**

$x{:}\in timer(1)$ $\qquad$ $x{:}\in timer(2)$

z is assigned *true* after 1 time unit.

Parallel or

Let sites M and N return booleans. Compute their parallel or.

$$z :\in ift(x) \mid ift(y) \mid or(x, y)$$

where

$$x :\in M$$
$$y :\in N$$

Similarly, evaluate any function f of the form

$$f(x, y) = \begin{cases} p(x) & \text{if } c(x) \\ q(y) & \text{if } d(y) \\ r(x, y) & \text{otherwise} \end{cases}$$

Eight queens

- configuration: placement of queens in the last i rows.
- Represent a configuration by a list of integers j, $0 \le j \le 7$.
- Valid configuration: no queen captures another.
- $check(x{:}xs)$: Given xs valid, return

  $x : xs$, if it is valid

  remain silent, otherwise.

Eight queens; Contd.

$$
\begin{array}{l}
let([\,]) \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle \\
\gg \ \langle check(0:\theta) \mid check(1:\theta) \mid check(2:\theta) \cdots \mid check(7:\theta)\rangle
\end{array}
$$

$$
\begin{array}{l}
let([\,]) \gg \langle \gg i : 0 \le i \le 7 : \\
\qquad\qquad \langle \mid j : 0 \le j \le 7 : \ check(j:\theta)\rangle \\
\qquad \rangle
\end{array}
$$

Local object

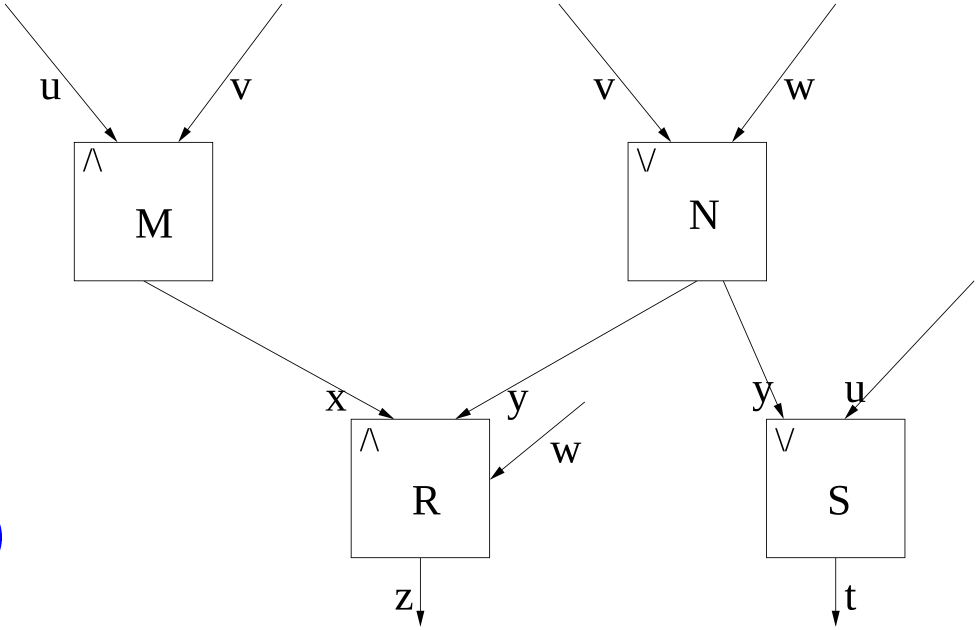
- Call sites M, N and R.
- Terminate after receiving two response.

Object $count$ with integer state. Initially, 0.

- $count.incr$ increments state;
- returns a signal if state ≥ 2, otherwise, remains silent.

$$
\begin{aligned}
& c{:}\in \\
& \quad\; M \gg count.incr \\
& \quad | \; N \gg count.incr \\
& \quad | \; R \gg count.incr
\end{aligned}
$$

And-Or graph



r:$\in$ $let(z, t)$

where

z:$\in$ $R(x, y, w)$

t:$\in$ $S(y) \mid S(u)$

where

x:$\in$ $M(u, v)$

y:$\in$ $N(v) \mid N(w)$

Airline

- Return any quote, from A or B, provided it is below 300.
- If neither quote is below 300, then return the cheapest quote or any quote available by time t.
- If no quote is available by t, return ∞.

Min returns the minimum of its argument values.

$threshold(x)$ returns x if x is below 300; silent otherwise.

$$z :\in threshold(x) \mid threshold(y) \mid Min(x, y)$$

where

$$x :\in A \mid timer(t, \infty)$$
$$y :\in B \mid timer(t, \infty)$$

Workflow: Visit Coordination

- $Email(p, s)$: contact p with dates s; response is date d from s.
- $Hotel(d)$: booking from hotel.
- $Airline(d)$: booking from airline.
- $Ack(p, t)$: similar to $Email$; response is an acknowledgment.
- $Confirm(t)$: confirm reservation t (for hotel or airline).
- $Room(d)$: reserve room for d. Response q: room number, time.
- $Announce(p, q)$: announce the lecture.
- $AV(q)$: contact technician with room and time information in q.

Workflow; Contd.

$z{:}\in let(b)$ $\gg$ $let(c,e)$ $\gg$ $let(u,v)$

where (for $let(b)$)

$b{:}\in Ack(p,h,f)$

where (for $let(c,e)$)

$c{:}\in Confirm(h)$
$e{:}\in Confirm(f)$

where (for $let(u,v)$)

$u{:}\in Announce(p,q)$
$v{:}\in AV(q)$

where

$q{:}\in Room(d)$

where

$h{:}\in Hotel(d)$
$f{:}\in Airline(d)$

where

$d{:}\in Email(p,s)$

Interrupt handling

- Orc statement can not be directly interrupted.
- $Interrupt$ site: a monitor.
- $Interrupt.set$: to interrupt the Orc statement
- $Interrupt.get$: responds after $Interrupt.set$ has been called.

$$z{:}\in f$$

is changed to

$$z{:}\in f \mid Interrupt.get$$

Processing Interrupt

$$z :\in \{\ f(x,y) \quad \textbf{where}\quad x :\in g,\ \ y :\in h\ \}$$

If f is interrupted, call M and N with parameters x and y, respectively, to cancel the effects of g and h.

$$z :\in\ Normal(t)\ \mid\ Interr(t) \gg let(X, Y)$$

where

$$X :\in M(x)$$
$$Y :\in N(y)$$

where

$$t\ :\in f(x,y)\ \mid\ Interrupt.get$$

where

$$x\ :\in g$$
$$y\ :\in h$$

Phase Synchronization

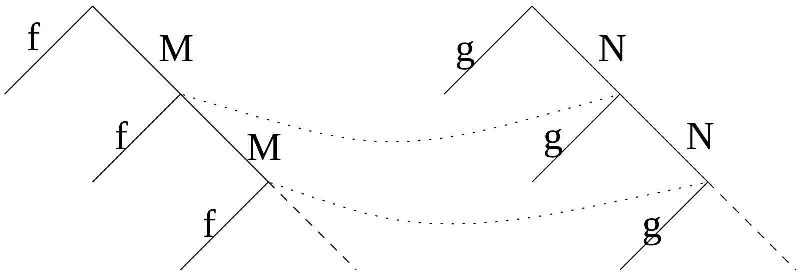
Process starts its $(k+1)^{th}$ phase only after all processes have completed their k^{th} phases.

Consider $M \gg f$ and $N \gg g$.

$$
\begin{array}{l}
\{let(x,y) \\
\quad\quad \textbf{where} \\
\quad\quad\quad\quad x{:}\in M \\
\quad\quad\quad\quad y{:}\in N\} \\
\gg \\
\quad fst(\theta) \gg f \;\mid\; snd(\theta) \gg g
\end{array}
$$

Phase Synchronization; Contd.

Synchronize $M^* \gg f$ and $N^* \gg g$.



$\{let(x, y)$

where

$x{:}\in M$

$y{:}\in N\}^*$

$\gg$

$fst(\theta) \gg f \mid snd(\theta) \gg g$

Heat transfer computation over a grid

- Value x_{ij} at point (i, j) in a phase is the average of its neighbors' values in the previous phase.
- Site $average$ returns the average of its arguments.
- Site $converge$ returns its argument value if the values have converged sufficiently, otherwise, it remains silent.

$$
\begin{array}{l}
z{:}\in \{let(x) \\
\qquad \textbf{where} \\
\qquad\qquad \langle \forall i, j :: \; x_{ij}{:}\in average(\theta_{i-1,j}, \theta_{i+1,j}, \theta_{i,j-1}, \theta_{i,j+1}) \rangle \\
\quad \}^{*} \\
\gg \; converge(\theta)
\end{array}
$$

Status of the Work

- Work extended to concurrency: Joint paper with Tony Hoare.
- Implemented under Java host language.
- A library of sites being built.
- Program Structuring:

  write expressions in more understandable ways

  introduce data and methods on them

  allow recursion.
- Programming Large Distributed Applications