

Meadow Use Cases



August 12, 2014

Contents

1	Topologies	2
1.1	Pair	2
1.2	Chain	2
1.3	Ring	2
1.4	Clique	2
1.5	Tree	2
1.6	Systolic array	2
2	Classic Problems	2
2.1	Producer-consumer	2
2.1.1	One producer, one consumer	2
2.2	Readers-writers	2
2.3	Dining philosopher	2
3	Hadoop style flow	2
3.1	Coroutine	2
3.2	Ping-ping	2
3.3	Fork-join: Divide-and-conquer	2
3.4	Fork-join-any	3
3.5	Fork-join-none	3
3.6	Map-reduce	3
3.7	Arbitrary graph	3
4	Workflow patterns	3
4.1	Workflow data patterns	3
4.2	Workflow control patterns	3
4.3	Workflow resource patterns	3
4.4	Workflow exception handling patterns	3
5	Service interaction patterns	3
6	Device selection patterns	3
6.1	Roles: Any device in the group	3

7	Communication patterns	4
7.1	Broadcasting	4
7.2	Multicasting	4
8	Data access/transfer patterns	4
8.1	Pipes	4
8.2	Blackboards	4
8.3	Shared variable	4
8.4	Streaming	4
9	Applications	4
9.1	Messenger	4
9.2	Chatting	4

1 Topologies

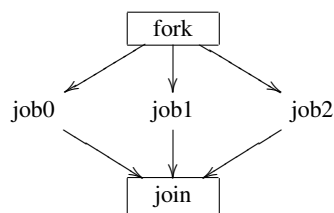
- 1.1 Pair
- 1.2 Chain
- 1.3 Ring
- 1.4 Clique
- 1.5 Tree
- 1.6 Systolic array

2 Classic Problems

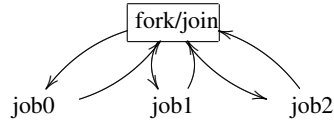
- 2.1 Producer-consumer
 - 2.1.1 One producer, one consumer
- 2.2 Readers-writers
- 2.3 Dining philosopher

3 Hadoop style flow

- 3.1 Coroutine
- 3.2 Ping-ping
- 3.3 Fork-join: Divide-and-conquer



An easy way to handle this is to see two points, **fork** and **join** as the identical point.



That is, **fork-join** distributes the same token to three nodes and continue following the flow graph when all three tokens come back.

```
process ForkJoin(dev0, dev1, dev2) {  
  function start() {  
    fork {  
      dev0.job0();  
      dev1.job1();  
      dev2.job2();  
    } join (?);  
  }  
}
```

3.4 Fork-join-any

3.5 Fork-join-none

3.6 Map-reduce

3.7 Arbitrary graph

4 Workflow patterns

4.1 Workflow data patterns

4.2 Workflow control patterns

4.3 Workflow resource patterns

4.4 Workflow exception handling patterns

5 Service interaction patterns

6 Device selection patterns

6.1 Roles: Any device in the group

Sometimes, we don't need some very specific device for a given role. Any device, which can satisfy the given role would suffice.

Situation In Seoul, we want to get information where the bus #51 that I'm waiting for is. Any bus with the number #51 is good that is close to me.

7 Communication patterns

7.1 Broadcasting

7.2 Multicasting

8 Data access/transfer patterns

8.1 Pipes

8.2 Blackboards

8.3 Shared variable

Synchronization is required.

8.4 Streaming

9 Applications

9.1 Messenger

9.2 Chatting

References

- [1] A. Barros, M. Dumas, and A. H. M. ter Hofstede. Service interaction patterns. In *Business Process Management*, volume 3649 of *Lecture Notes in Computer Science*, pages 302–318, 2005.
- [2] Hohpe and Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
- [3] N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow data patterns. QUT Technical Report FIT-TR-2004-01, Queensland University of Technology, 2004.
- [4] N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow resource patterns. BETA Working Paper Series WP 127, Eindhoven University of Technology, 2004.
- [5] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns : A revised view. BPM Center Report BPM-06-22, BPM Center, 2006.