3C03 Concurrency: Concurrent Architectures - Supervisor-Worker

Wolfgang Emmerich

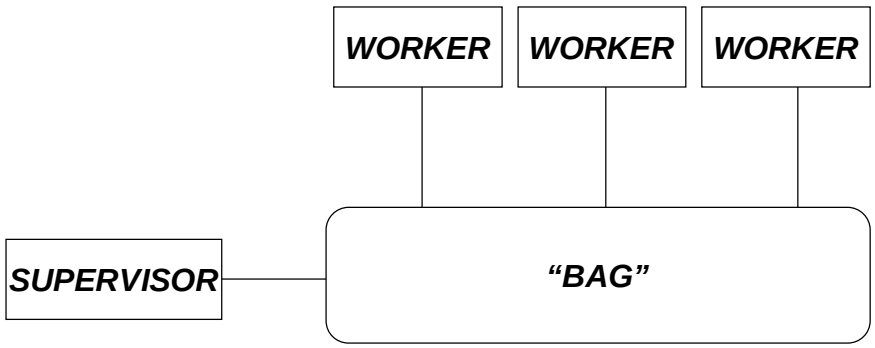
Outline

- Motivation
- Linda Tuple Spaces
- Modelling Tuple Spaces in FSP
- Implementing Tuple Spaces in Java
- Supervisor-Worker Model
- Supervisor-Worker Java Implementation

Motivation

- *Exploiting parallel execution on multiple processors*
- *Communication between different processors by a connector called “bag”*
- *Supervisor creates tasks and puts them into bag*
- *Workers pick tasks from bag and perform them*
- *Workers may themselves be supervisors*

Supervisor-Worker Architecture



Linda Tuple Spaces

- *Primitive for implementing "bag" connectors*
- *Tuple is a tagged data record*
- *Tuples are exchanged in tuple spaces using associative memory*
- *Available basic operations:*
  - `out("tag",expr1,…,exprn)`
  - `in("tag",field1,…,fieldn)`
  - `rd("tag",field1,…,fieldn)`
  - `inp("tag",field1,…,fieldn)`
  - `rdp("tag",field1,…,fieldn)`

Tuple Space Model

```
const N=2
set Tuples={any}
const False = 0
const True = 1
range Bool = False..True
TUPLE(T='any) = TUPLE[0],
TUPLE[i:0..N]=(
           when (i<N) out[T] -> TUPLE[i+1]
          |when (i>0) in[T] -> TUPLE[i-1]
          |when (i>0) inp[True][T] -> TUPLE[i-1]
          |when (i==0)inp[False][T] -> TUPLE[i]
          |when (i>0) rd[T] -> TUPLE[i]
          |rdp[i>0][T] -> TUPLE[i]).
||TUPLESPACE = forall [t:Tuples] TUPLE(t).
```

LTSA

Tuple Space Java Implementation

```
public interface  TupleSpace {
  //deposits data in tuple space
  public void out(String tag, Object data);
  //extracts object with tag from tuple space
  public Object in(String tag) throws
                          InterruptedException;
  //reads object with tag from tuple space
  public Object rd(String tag) throws
                          InterruptedException;
  //extracts object if avail else return null
  public Object inp(String tag);
  //read object if avail else return null
  public Object rdp(String tag);
}
```


Supervisor-Worker Algorithm

- ***Supervisor::***

  forall tasks do out("task",…) end

  forall results: in("result",…) end

  out("stop")

- ***Worker::***

  while not rdp("stop") do

  in("task",…)

  compute result

  out("result",…)

  end

Supervisor-Worker Model

LTSA

```
const N = 2
set Tuples = {task,result,stop}
set TupleAlpha =
        {{in,out,rd,rdp[Bool],inp[Bool]}.Tuples}
SUPERVISOR = TASK[1],
TASK[i:1..N] = (out.task ->
   if i<N then TASK[i+1] else RESULT[1]),
RESULT[i:1..N]= (in.result ->
   if i<N then RESULT[i+1] else FINISH),
FINISH = (out.stop->end->STOP)+TupleAlpha.
WORKER = (rdp[b:Bool].stop->
   if (!b) then (in.task->out.result->WORKER)
           else (end -> STOP) )+TupleAlpha.
END = (end ->ENDED), ENDED = (ended->ENDED).
||SUPERVISOR_WORKER=(supervisor:SUPERVISOR
  ||{redWork,blueWork}:WORKER
  ||{supervisor,redWork,blueWork}::TUPLESPACE
  ||END)/{end/{supervisor,redWork,blueWork}.end}.
```


Analysis of Supervisor-Worker Model

- *Trace to DEADLOCK:*

```
supervisor.out.task
supervisor.out.task
redWork.rdp.0.stop
redWork.in.task
redWork.out.result
supervisor.in.result
redWork.rdp.0.stop
redWork.in.task
redWork.out.result
supervisor.in.result
redWork.rdp.0.stop
supervisor.out.stop
```

Deadlock Free Algorithm

- ***Supervisor::***

  forall tasks:- out("task",...)

  forall results: in("result",...)

  out("stop")

- ***Worker::***

  while true do

  in("task",...)

  If value is stop then out("task",stop); exit

  compute result

  out("result",...)

Deadlock Free Model

```
set Tuples = {task,task.stop,result}
SUPERVISOR = TASK[1],
TASK[i:1..N] = (out.task ->
   if i<N then TASK[i+1] else RESULT[1]),
RESULT[i:1..N] = (in.result ->
if i<N then RESULT[i+1] else FINISH),
FINISH=(out.task.stop->end->STOP)+TupleAlpha.
WORKER=(in.task -> out.result -> WORKER
       |in.task.stop->out.task.stop->end->STOP
       )+ TupleAlpha.


progress={ended}
```

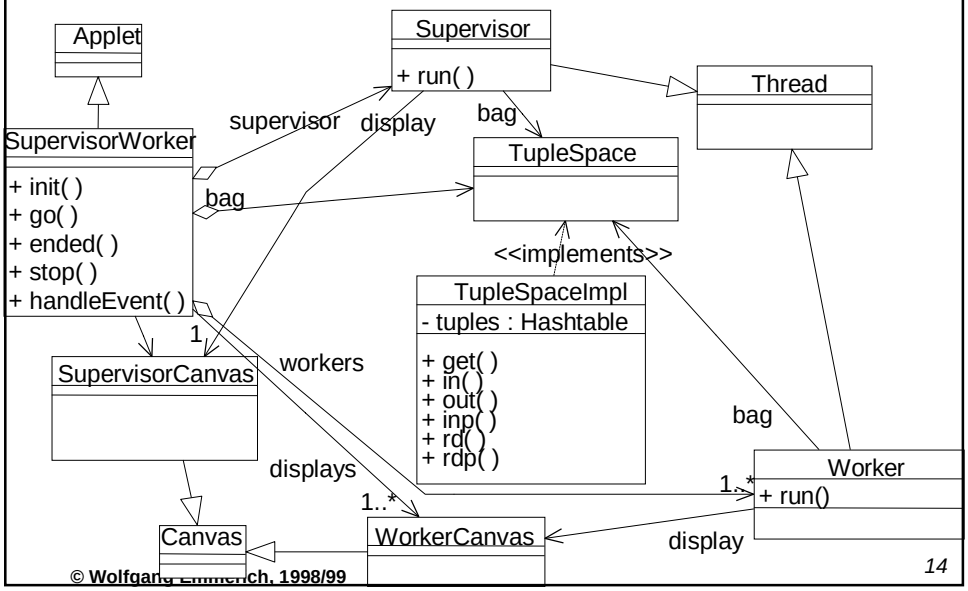
LTSA

Supervisor-Worker Example

- *Compute the area under a curve*
- *Approximate using rectangles*
- *Parallelize task by delegating computation of different rectangles to one of 4 workers*
- *Supervisor adds results computed by 4 workers*

Demo

Supervisor-Worker Example Design



Summary

- *Motivation*
- *Linda Tuple Spaces*
- *Modelling Tuple Spaces in FSP*
- *Implementing Tuple Spaces in Java*
- *Supervisor-Worker Model*
- *Supervisor-Worker Java Implementation*