Code Generation of Time Critical Synchronous Programs on the Kalray MPPA Many-Core architecture

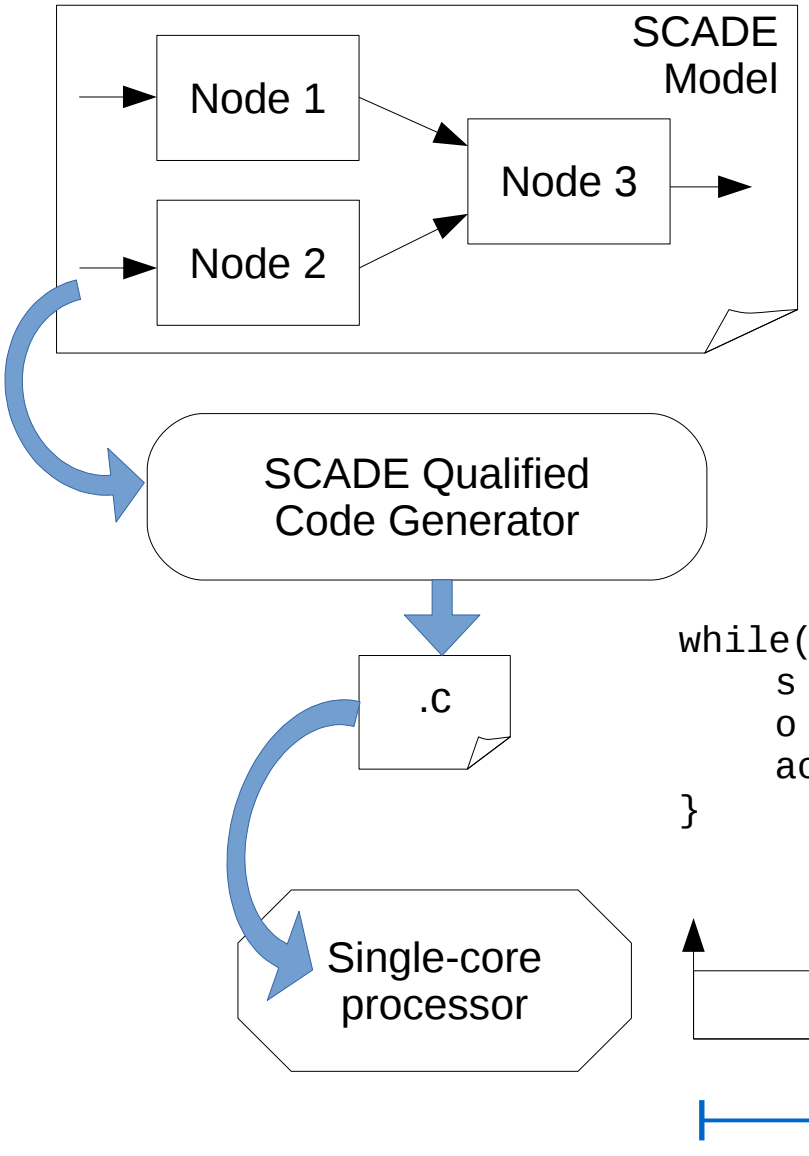
Amaury Graillat (PhD student)

Supervisors: Matthieu Moy, Pascal Raymond (Verimag)

Benoit Dinechin (Kalray)

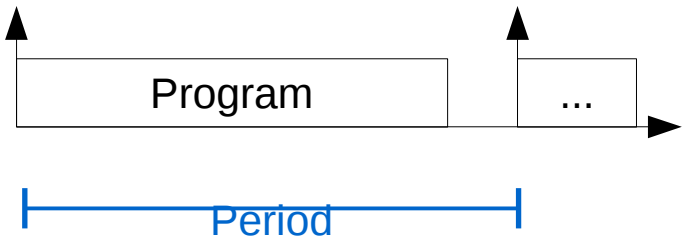
WATERS 2016, Tuesday 26th May 2016

Synchronous Languages: Traditional Approach



Formal semantics
Determinism

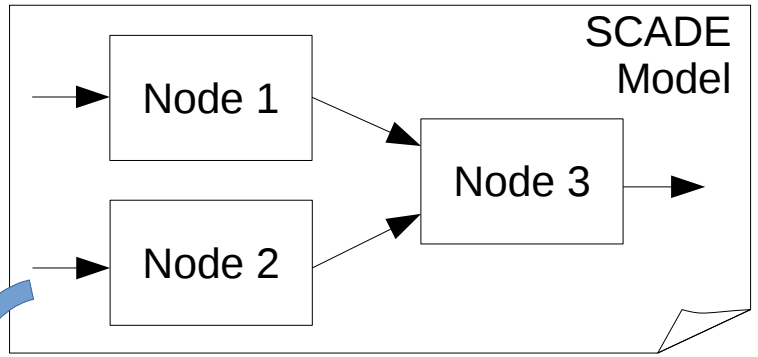
```
while(true) {
    s = sensors()
    o = compute(s)
    actuators(o)
}
```



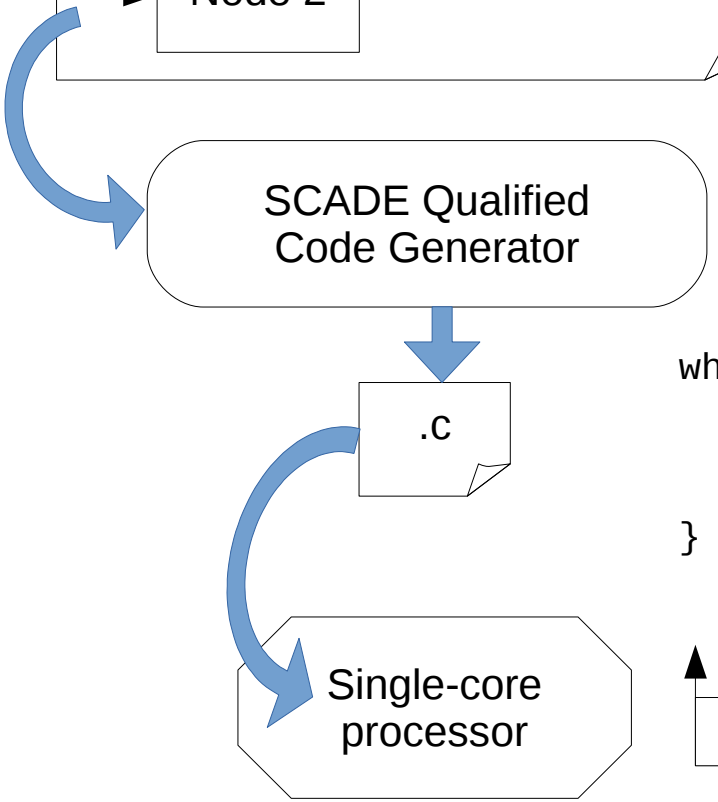
Synchronous languages

- Lustre, Heptagon, SCADE
- Industrial use: SCADE in Airbus A380

Synchronous Languages: Traditional Approach



Formal semantics
Determinism

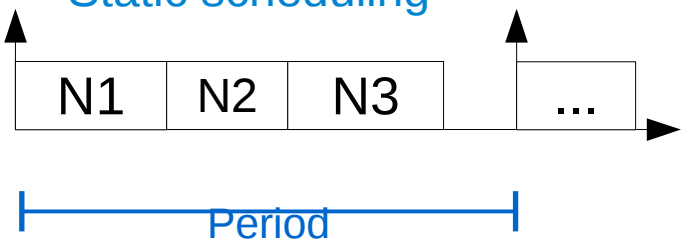



```
while(true) {
    s = sensors()
    o = compute(s)
    actuators(o)
}
```

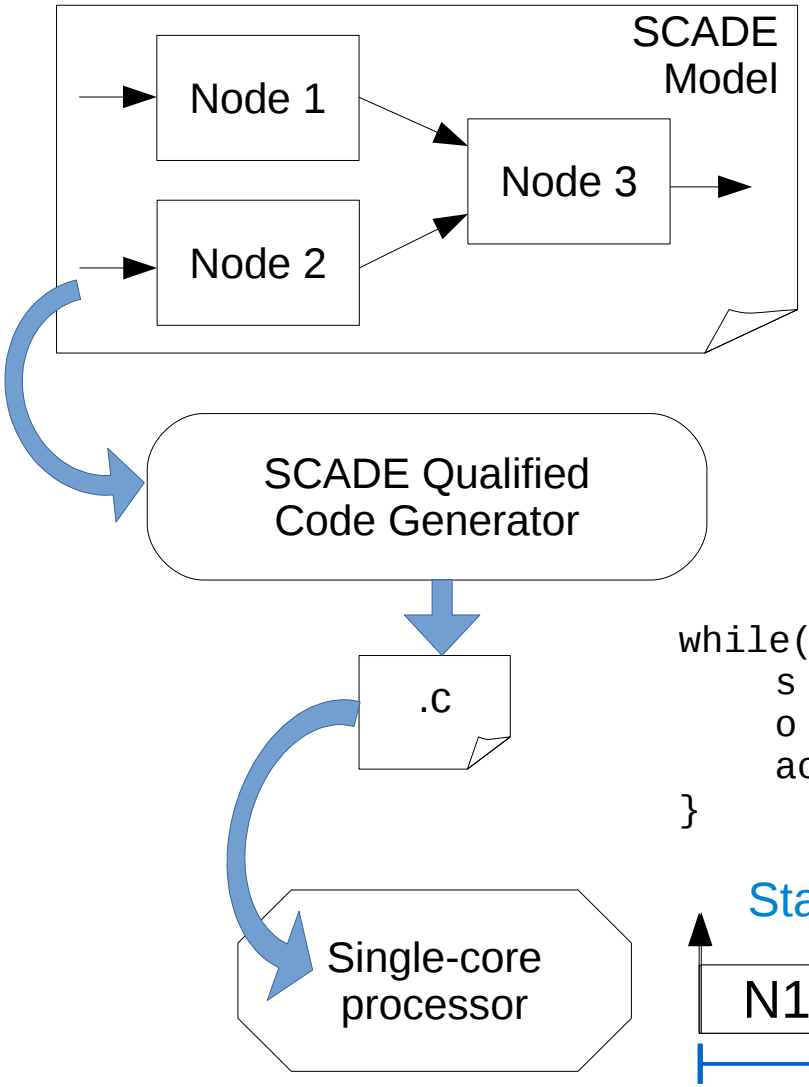
Static scheduling

```
while(true) {
    s = sensors()
    o1 = N1(s);
    o2 = N2(s);
    o = N3(o1,o2);
    actuators(o)
}
```

Static scheduling



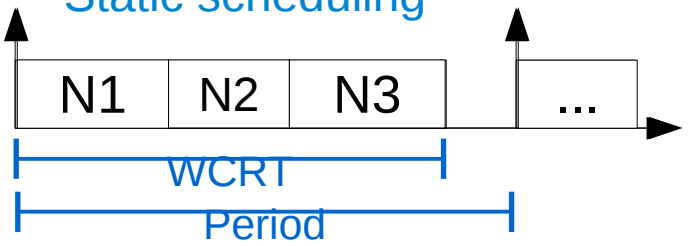
Synchronous Languages: Traditional Approach



Formal semantics
Determinism

```
while(true) {
    s = sensors()
    o = compute(s)
    actuators(o)
}
```

Static scheduling

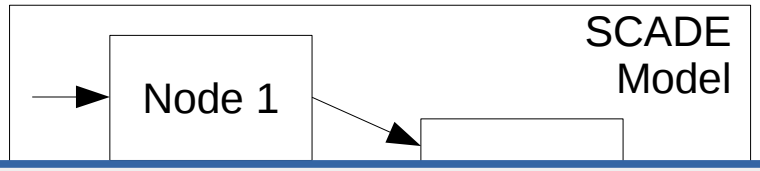


Synchronous languages

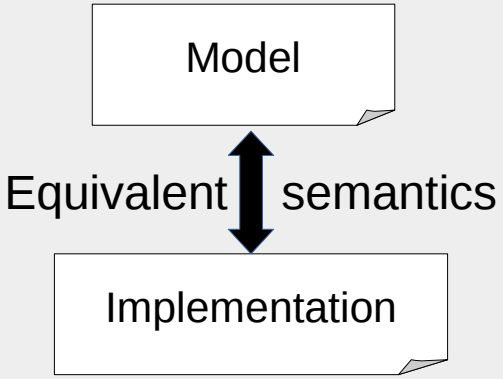
- Lustre, Heptagon, SCADE
- Industrial use: SCADE in Airbus A380

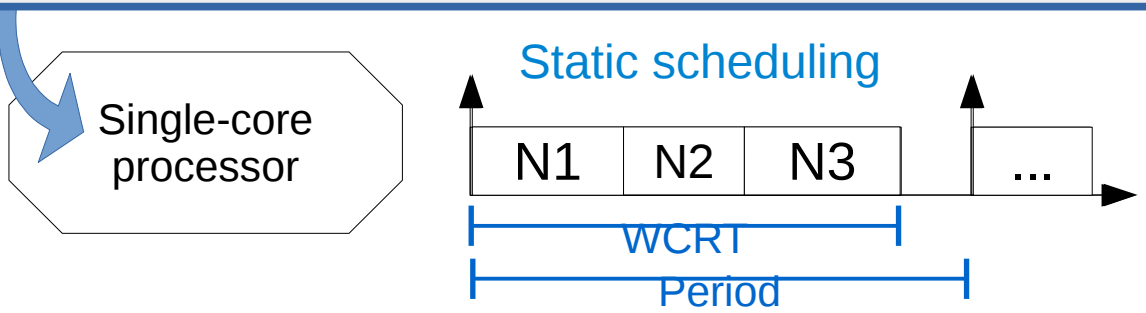
Real-time = Worst-Case Response Time < Period

Synchronous Languages: Traditional Approach



Formal semantics

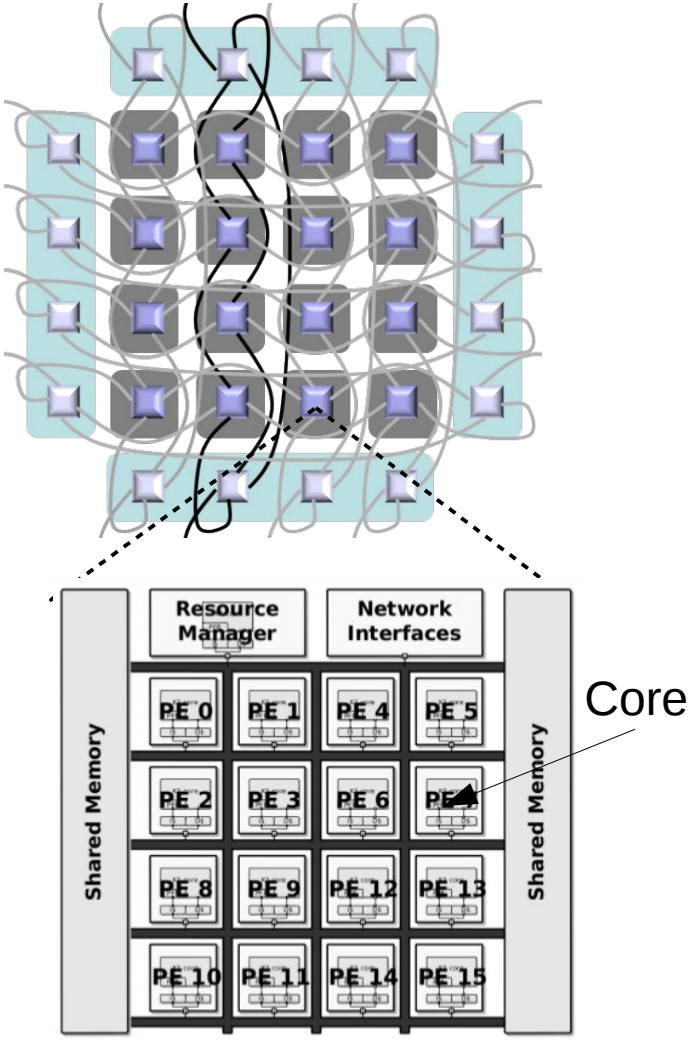




Real-time = Worst-Case Response Time < Period

Many-Core: example of the Kalray MPPA

"Several processors + network."



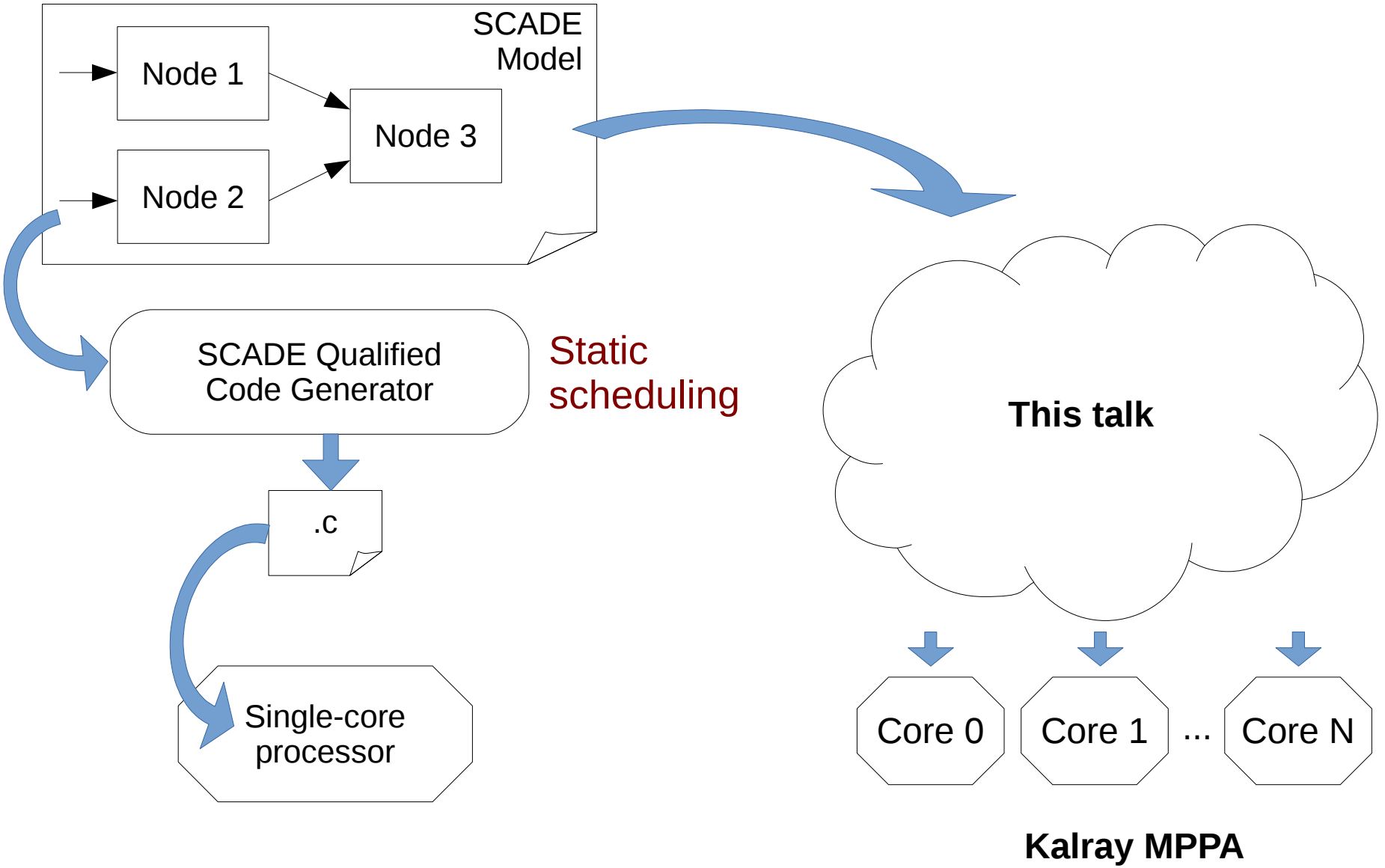
Cores:
- No complex branch prediction
- Only LRU caches

Cluster:
- Banked Shared-Memory (16*128ko)
- One round-robin for each bank access

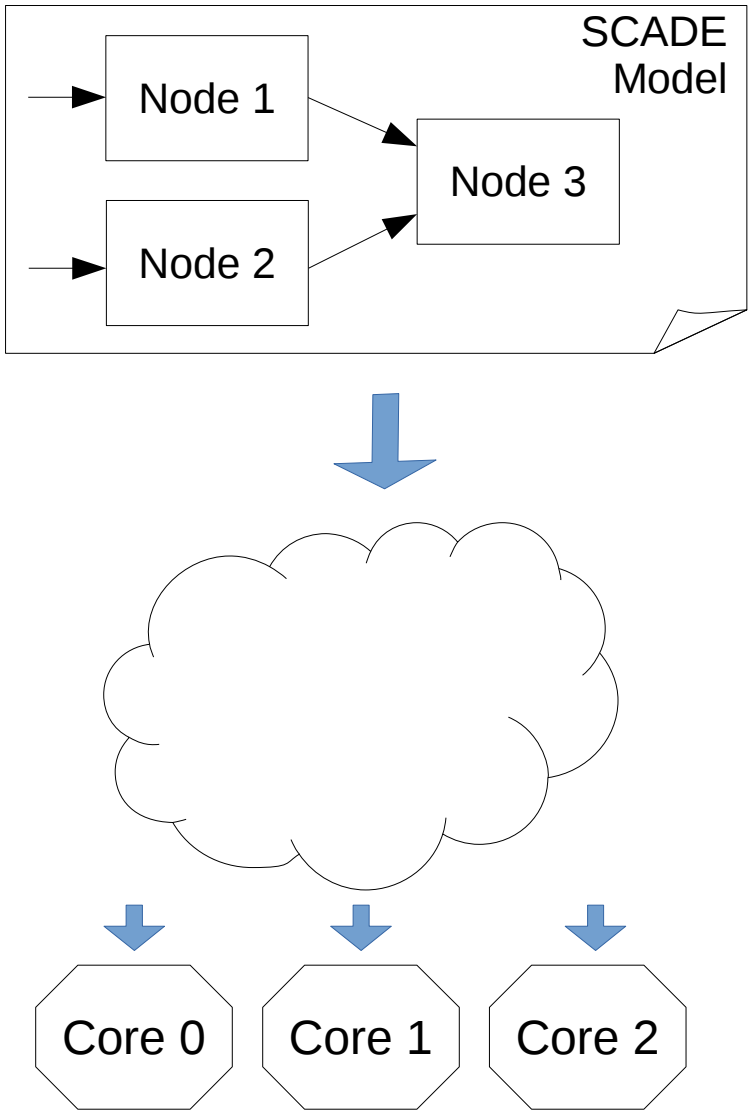
Network-on-Chip to connect clusters
- Bandwidth limiter (Network calculus)

In this talk: code generation whithin one cluster.

Synchronous Program on Many-Core



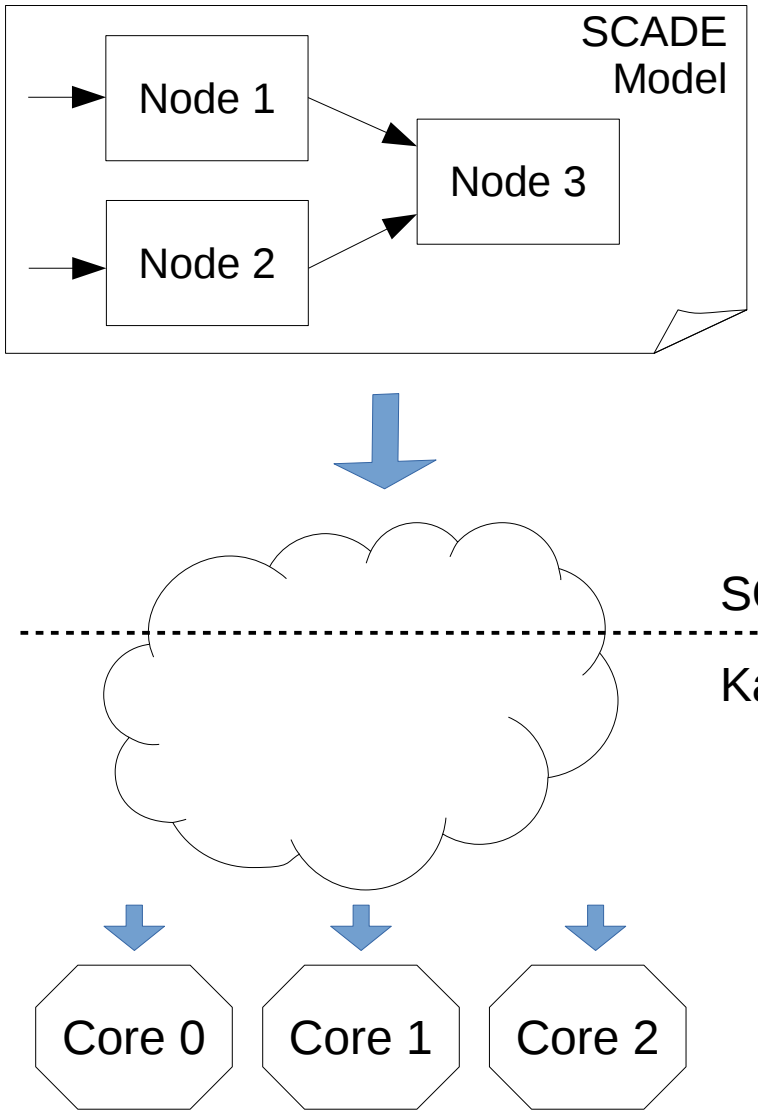
Issues



Kalray MPPA

- Split into tasks
- Mapping
- Scheduling
- Communication

Issues

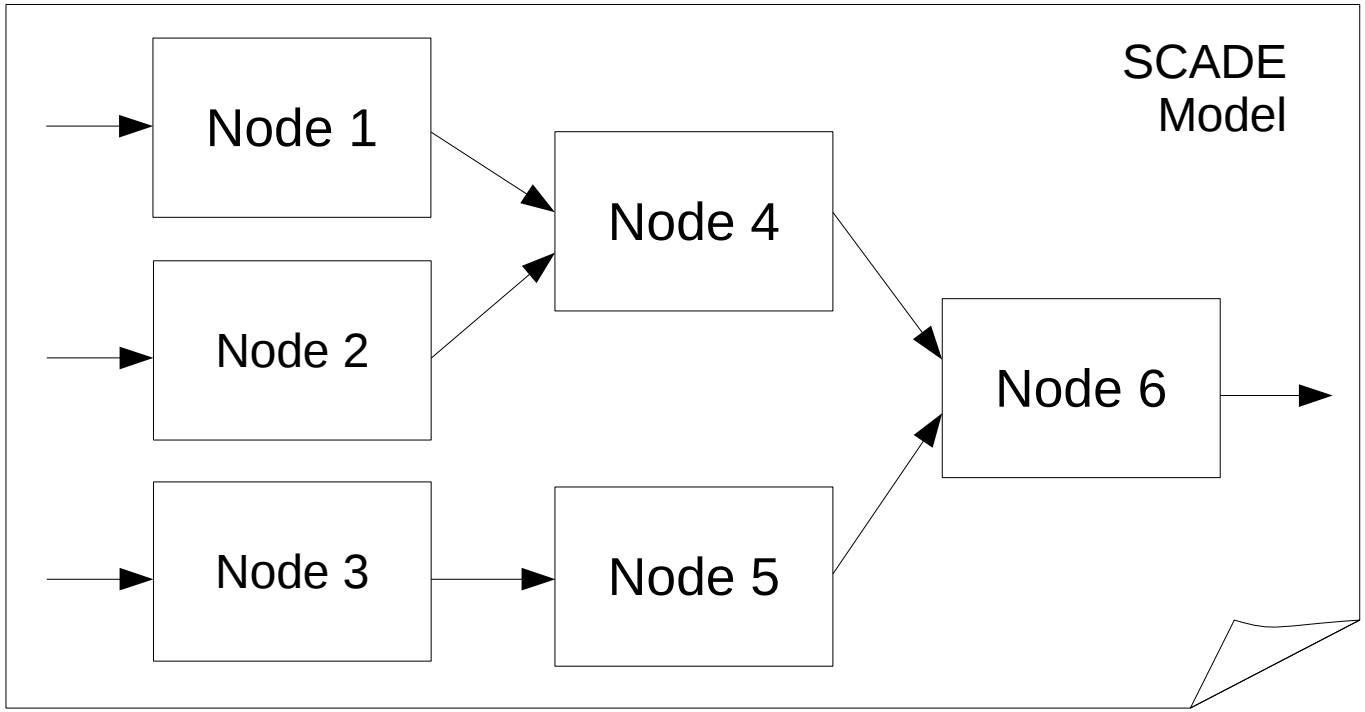


SCADE specific (Ansys)

Kalray specific

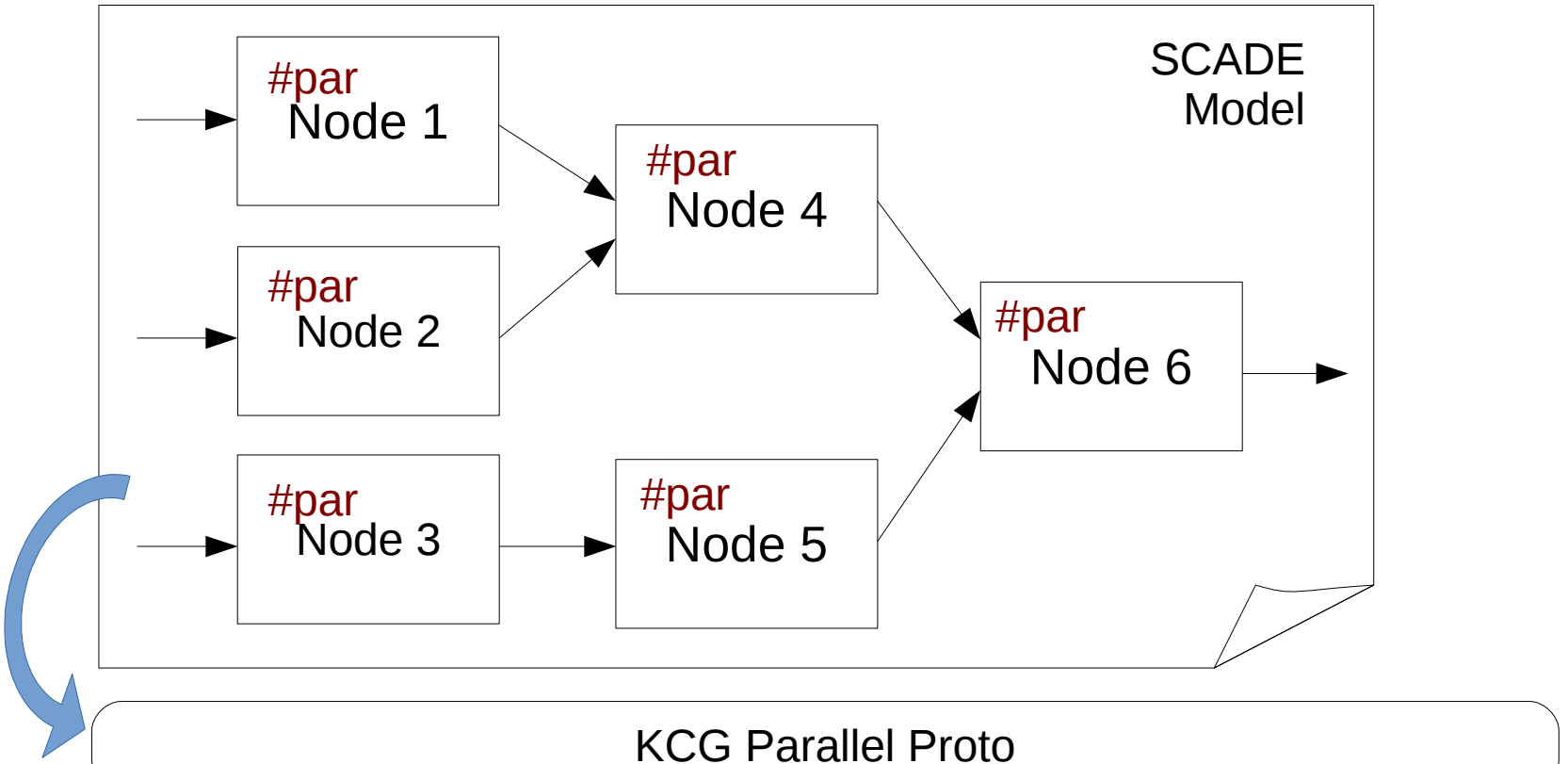
- Split into tasks
- Mapping
- Scheduling
- Communication

Split into Parallel Tasks



Split into Parallel Tasks

- KCG Parallel Prototype
- #par notation for developer



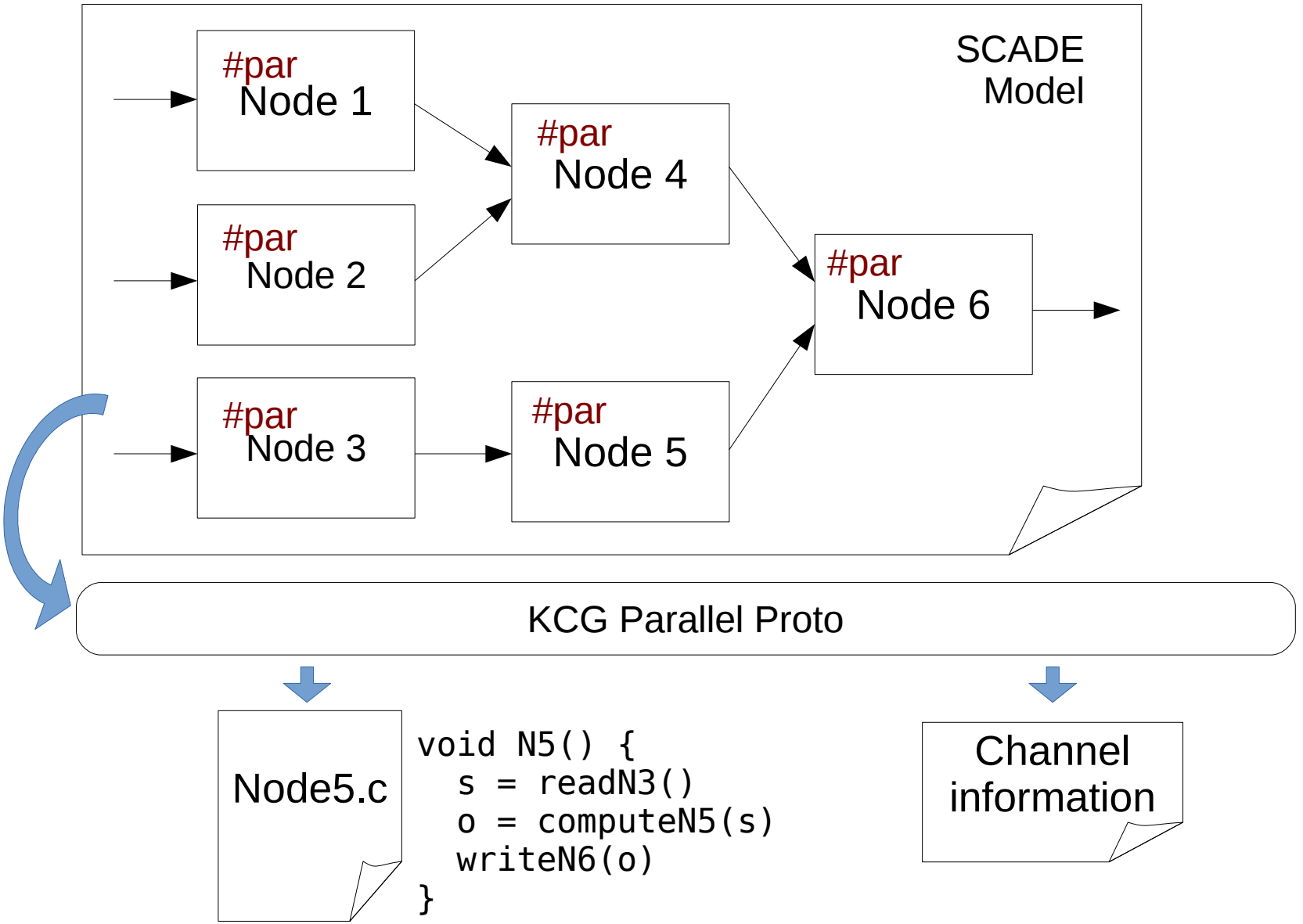
Node5.c

```
void N5() {
  s = readN3()
  o = computeN5(s)
  writeN6(o)
}
```

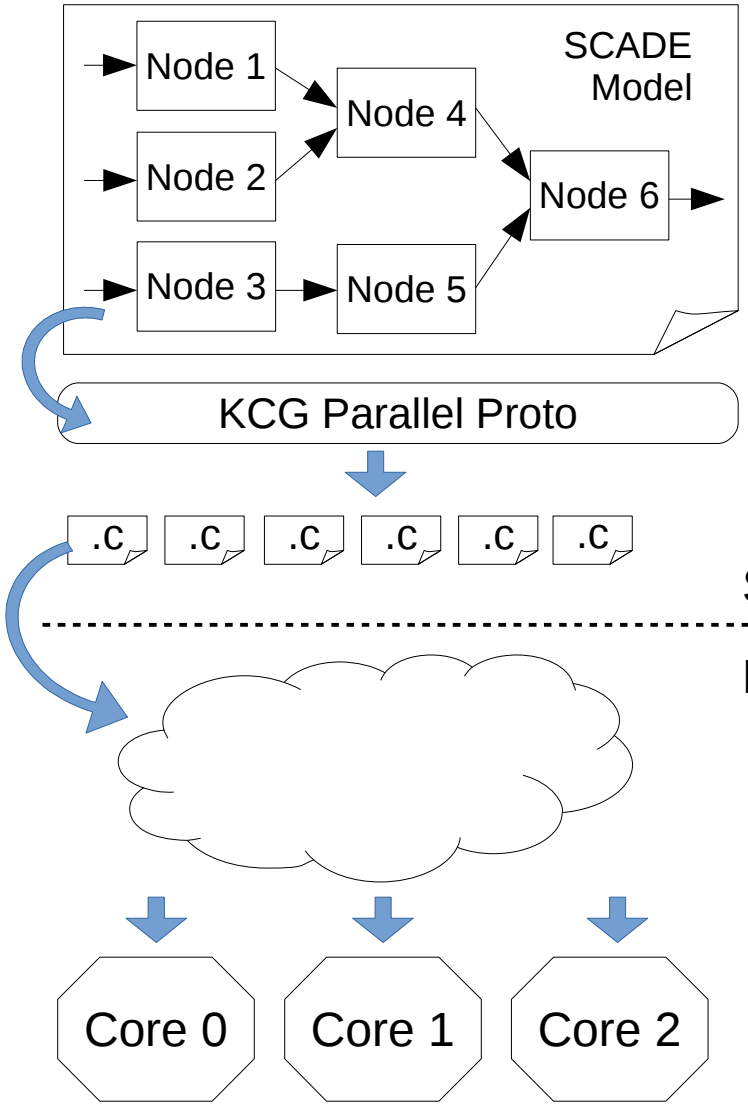
Well defined API.

Split into Parallel Tasks

- KCG Parallel Prototype
- #par notation for developer



Issues



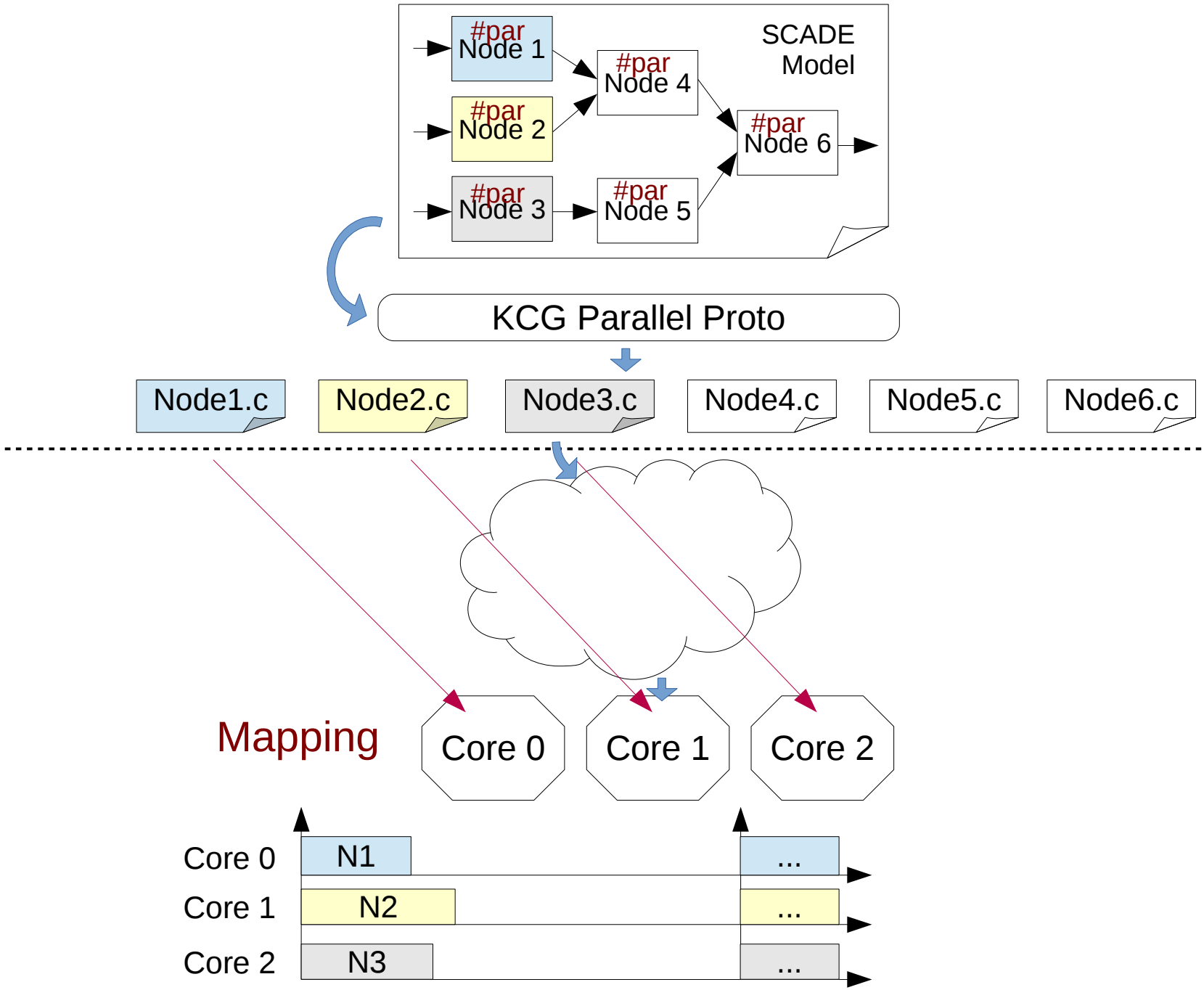
Kalray MPPA

SCADE specific

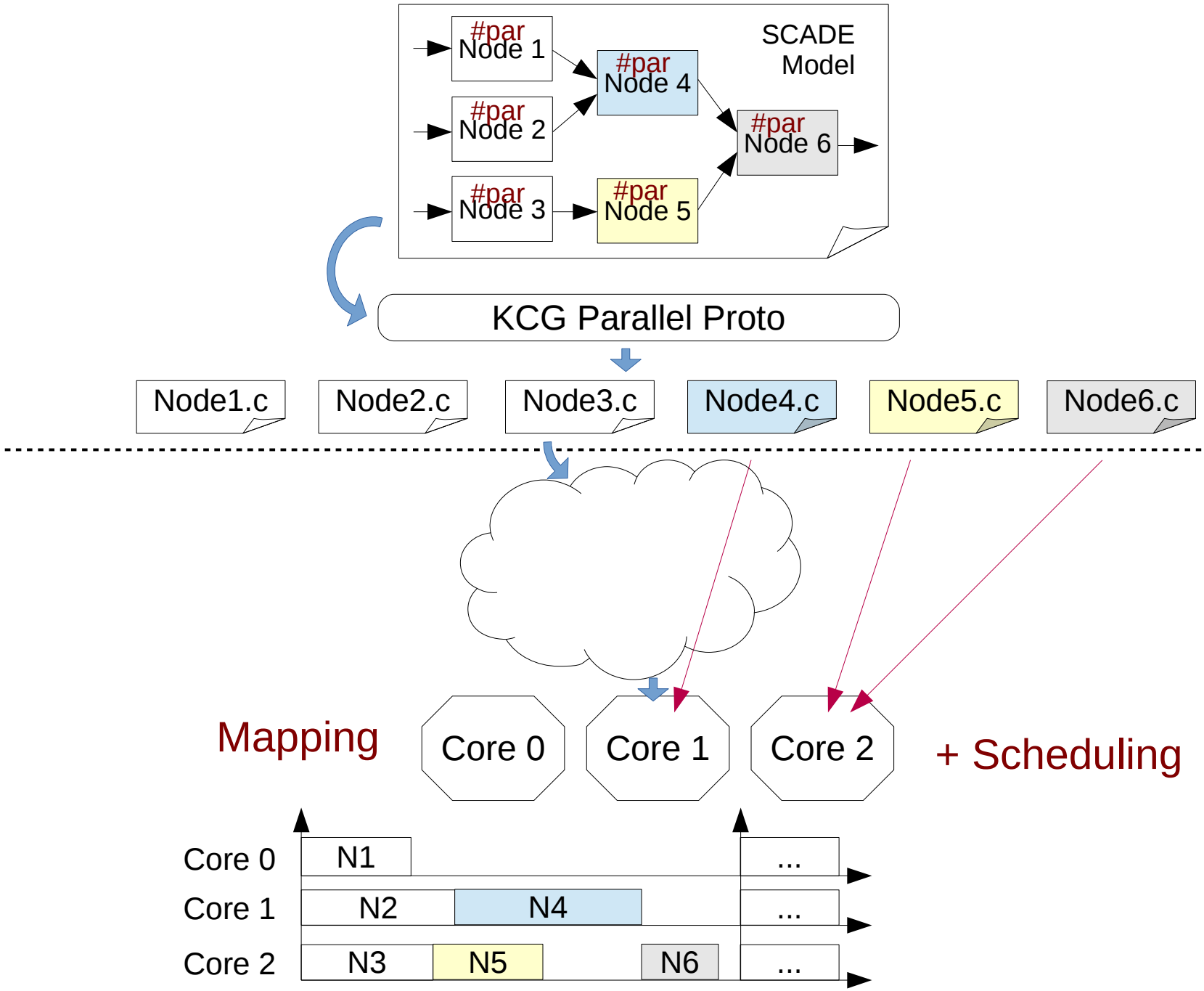
Kalray specific

- Split into tasks
- Mapping
- Scheduling
- Communication

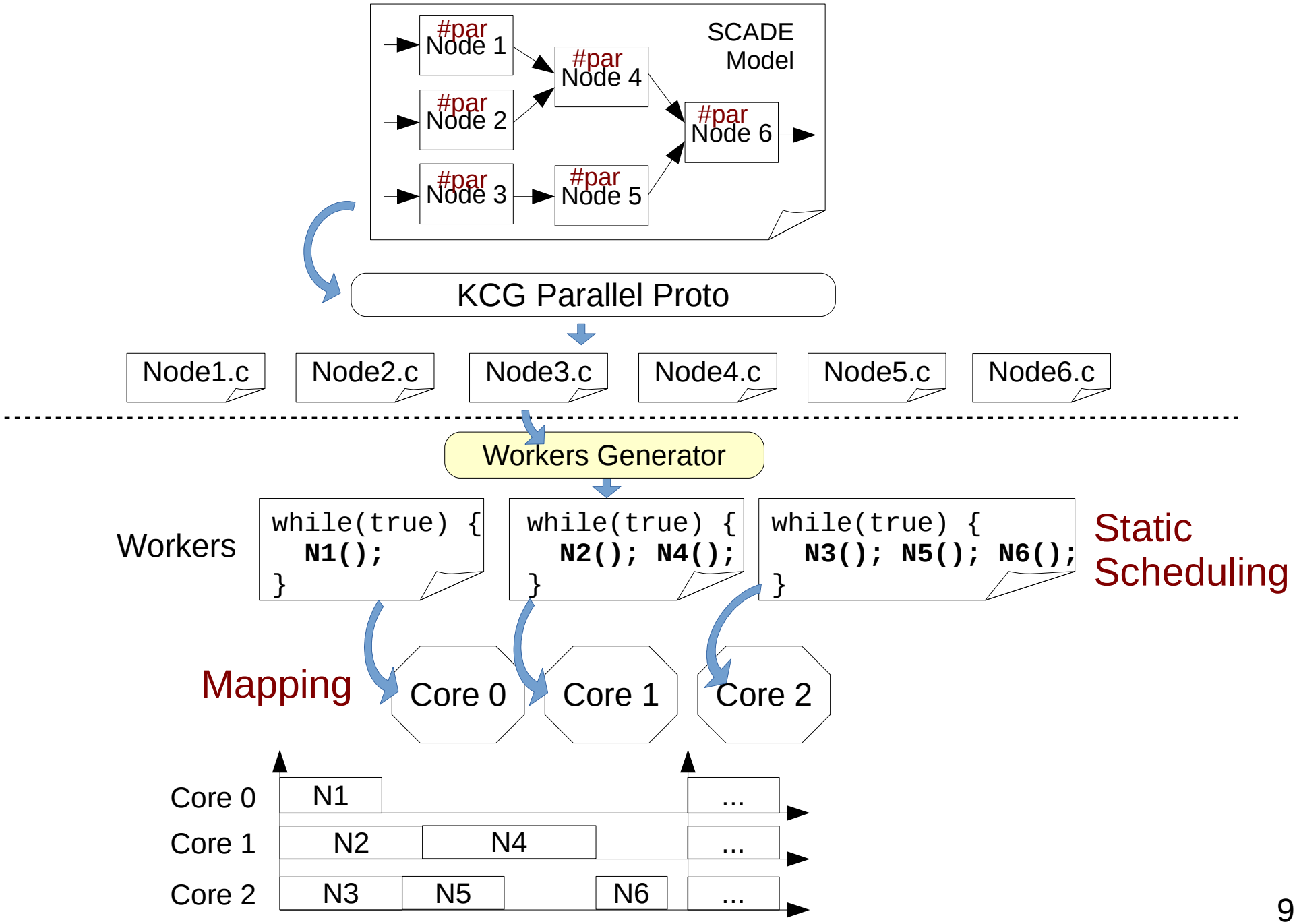
Mapping


#par
Node 1
#par
Node 2
#par
Node 3
#par
Node 4
#par
Node 5
#par
Node 6
SCADE
Model
KCG Parallel Proto
Node1.c
Node2.c
Node3.c
Node4.c
Node5.c
Node6.c
Mapping
Core 0
Core 1
Core 2
Core 0
Core 1
Core 2
N1
N2
N3
...
...
...

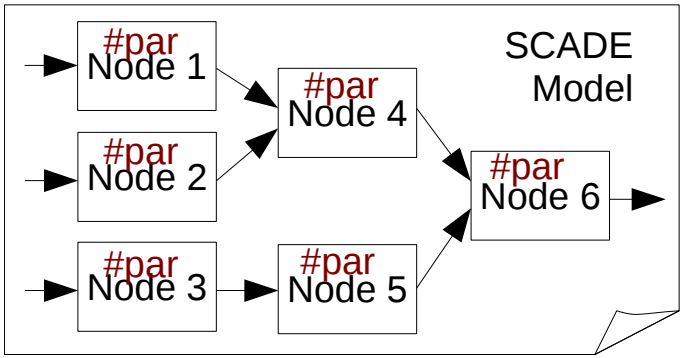
Mapping + Scheduling



Generate Worker



Static Scheduling

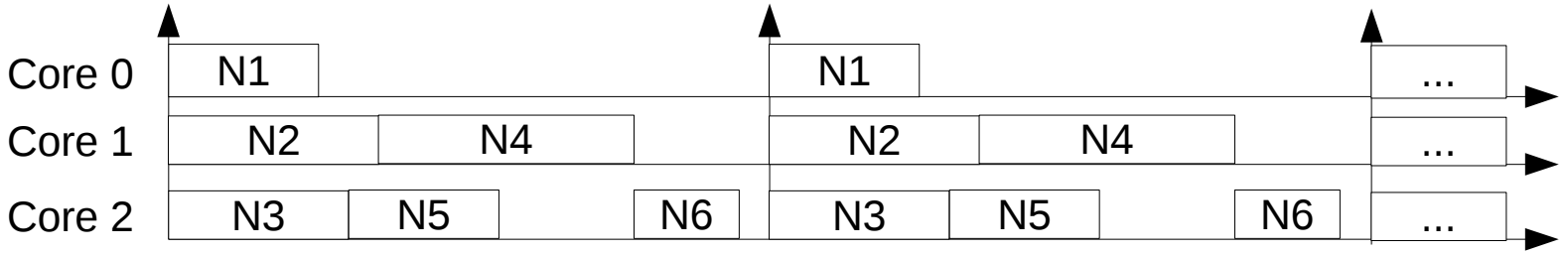


Workers

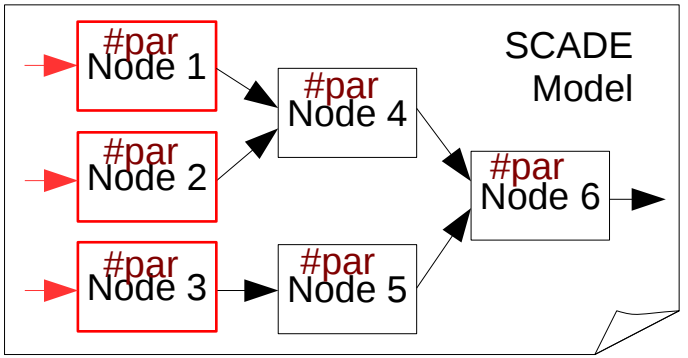
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

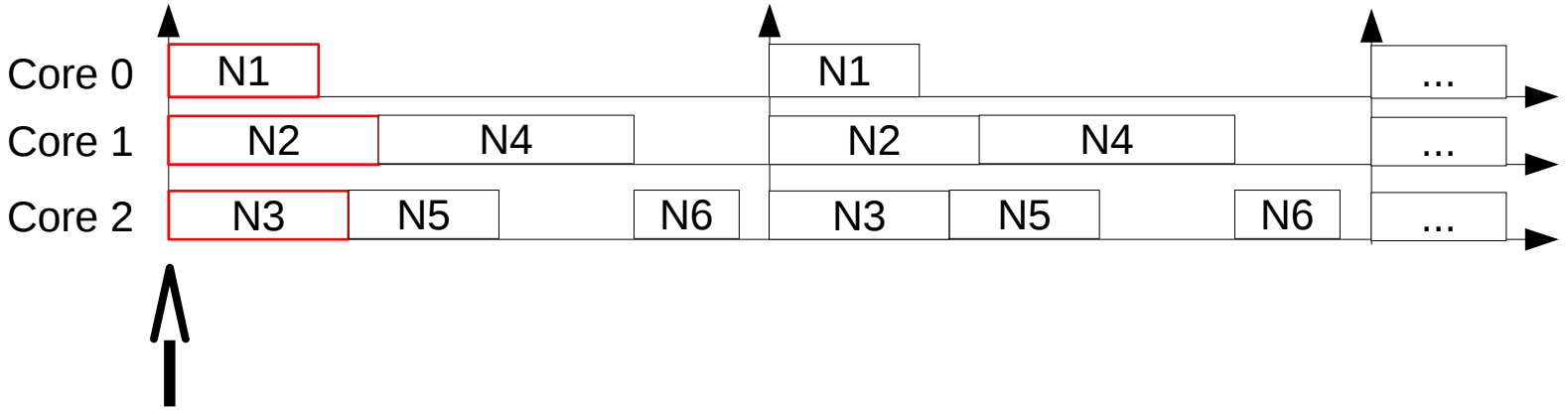


Workers

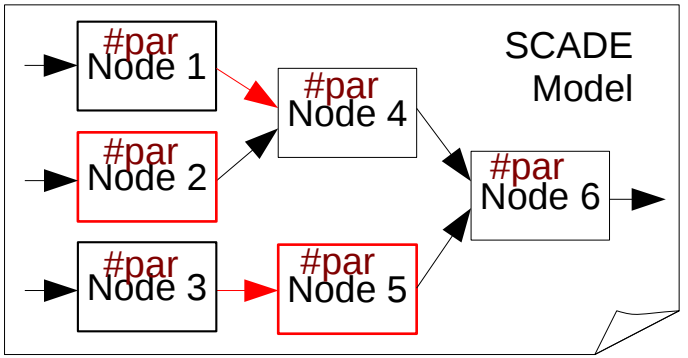
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

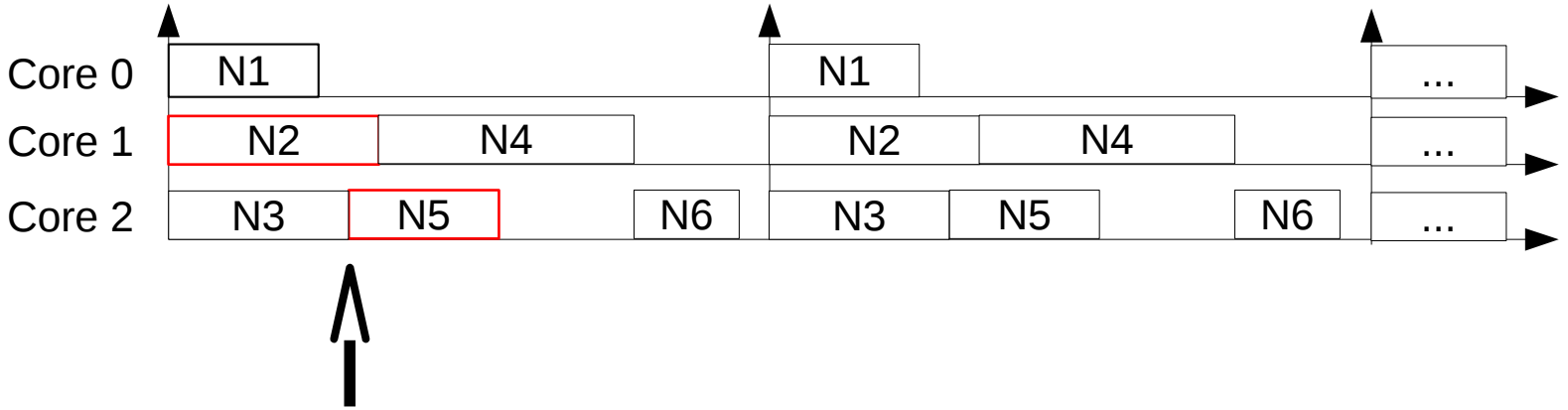


Workers

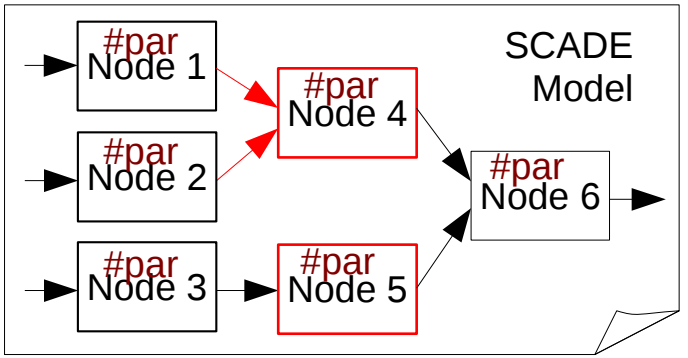
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

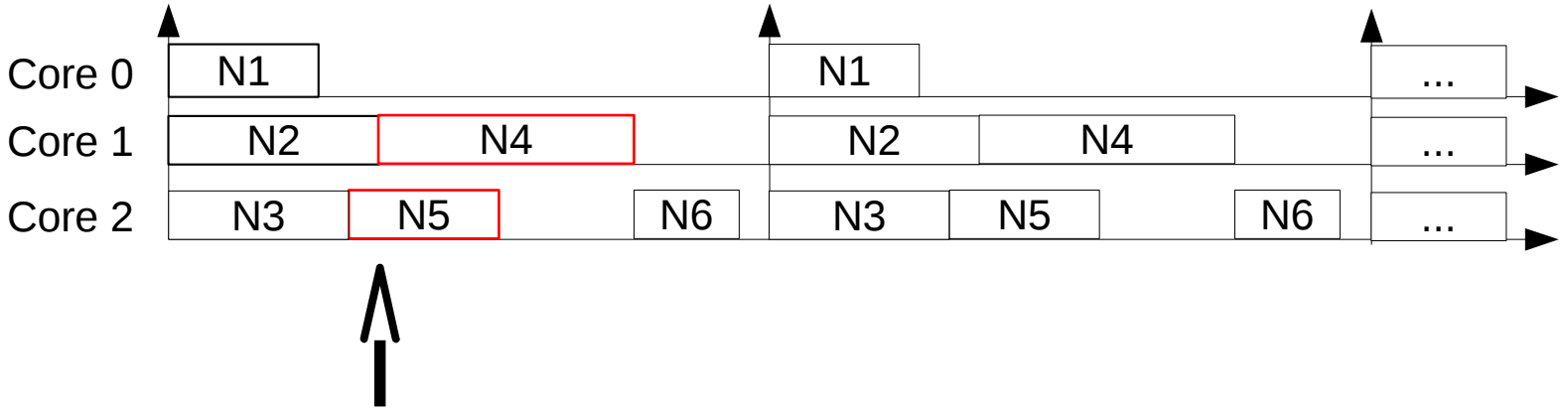


Workers

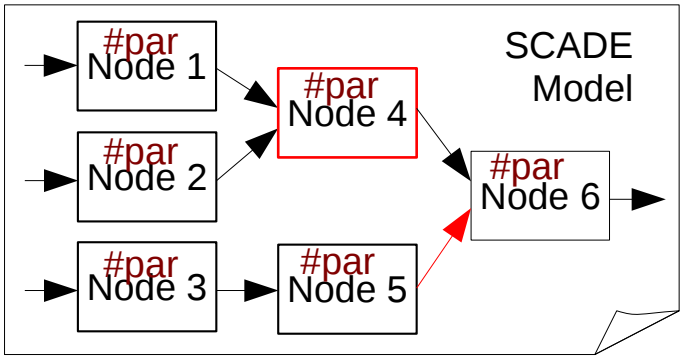
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

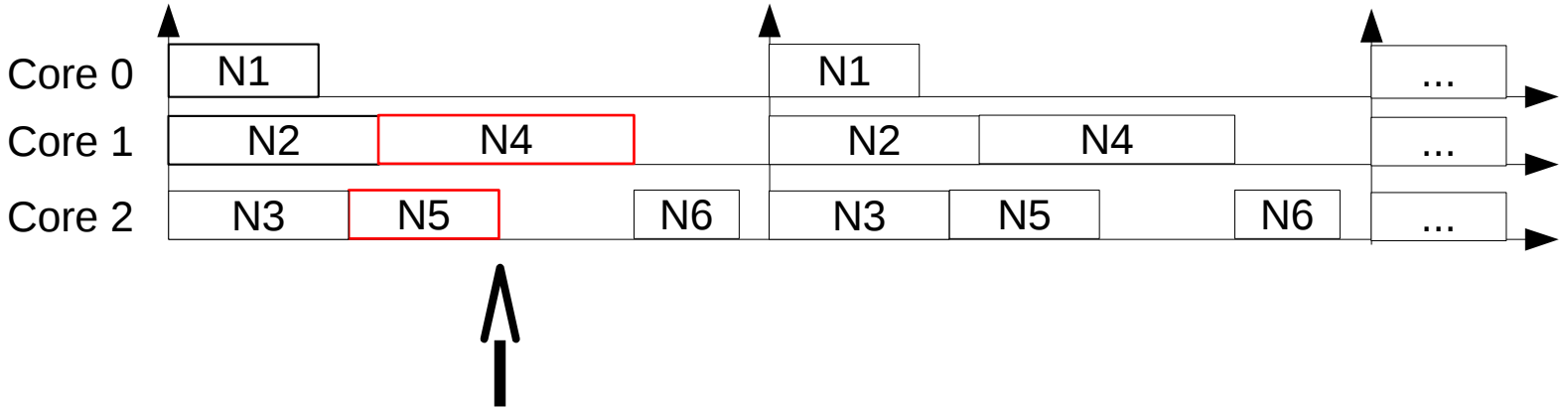


Workers

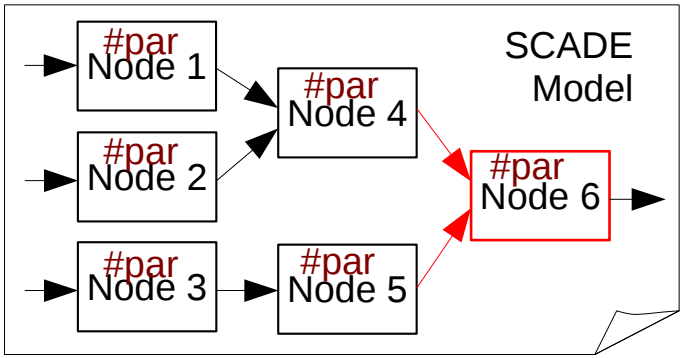
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

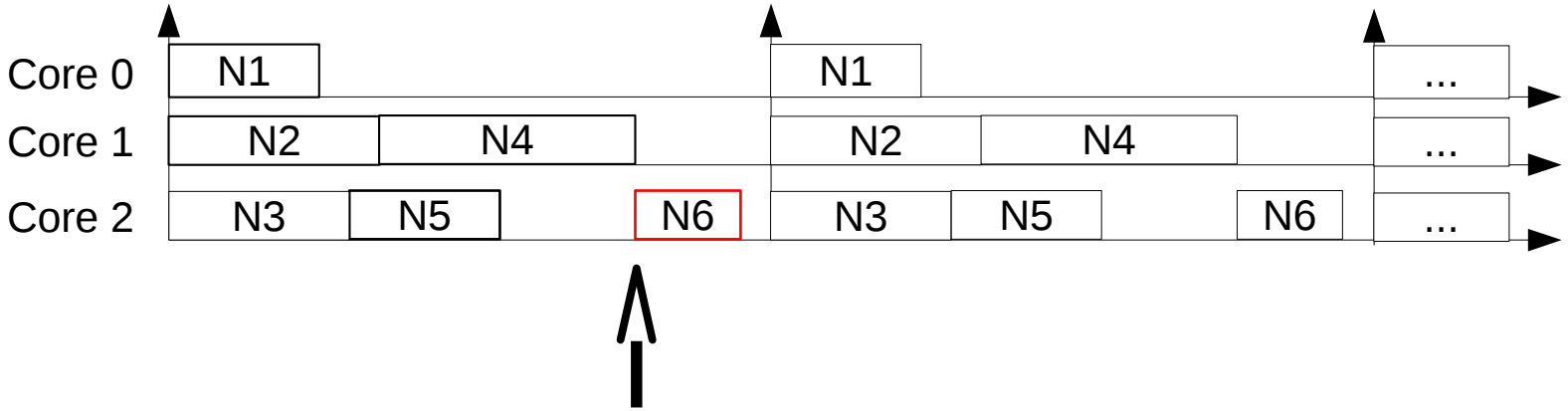


Workers

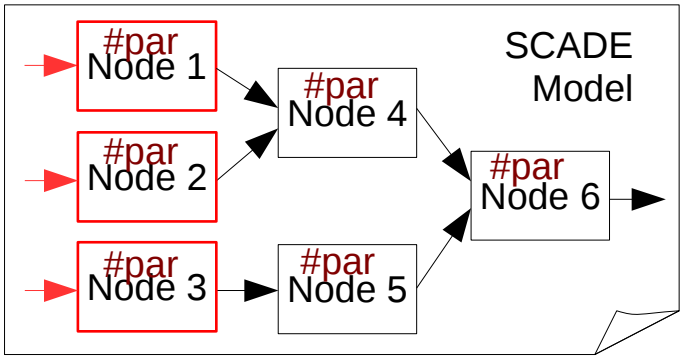
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```



Static Scheduling

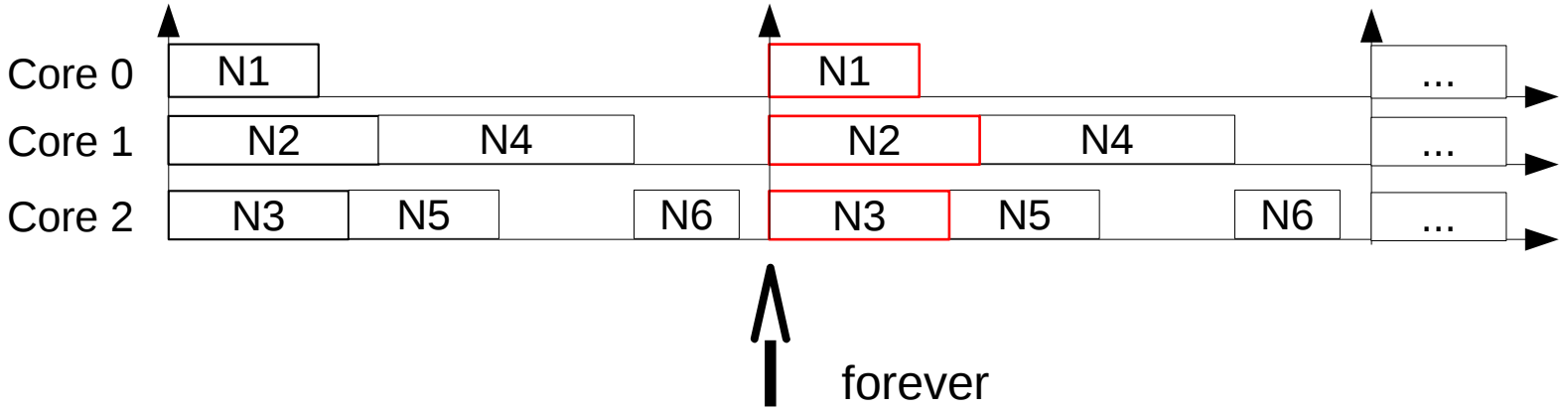


Workers

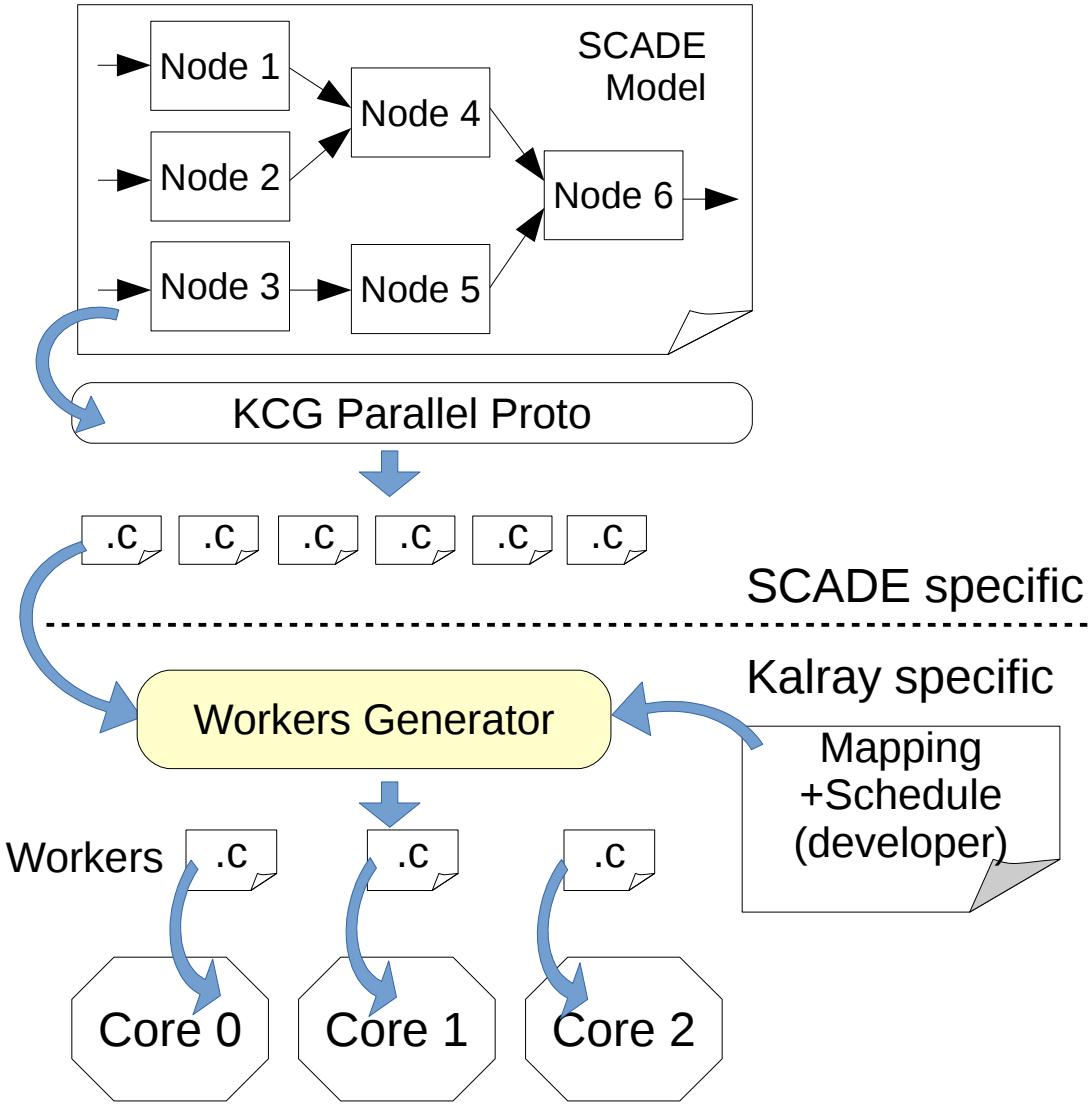
```
while(true)
{
  N1();
}
```

```
while(true)
{
  N2();
  N4();
}
```

```
while(true)
{
  N3();
  N5();
  N6();
}
```

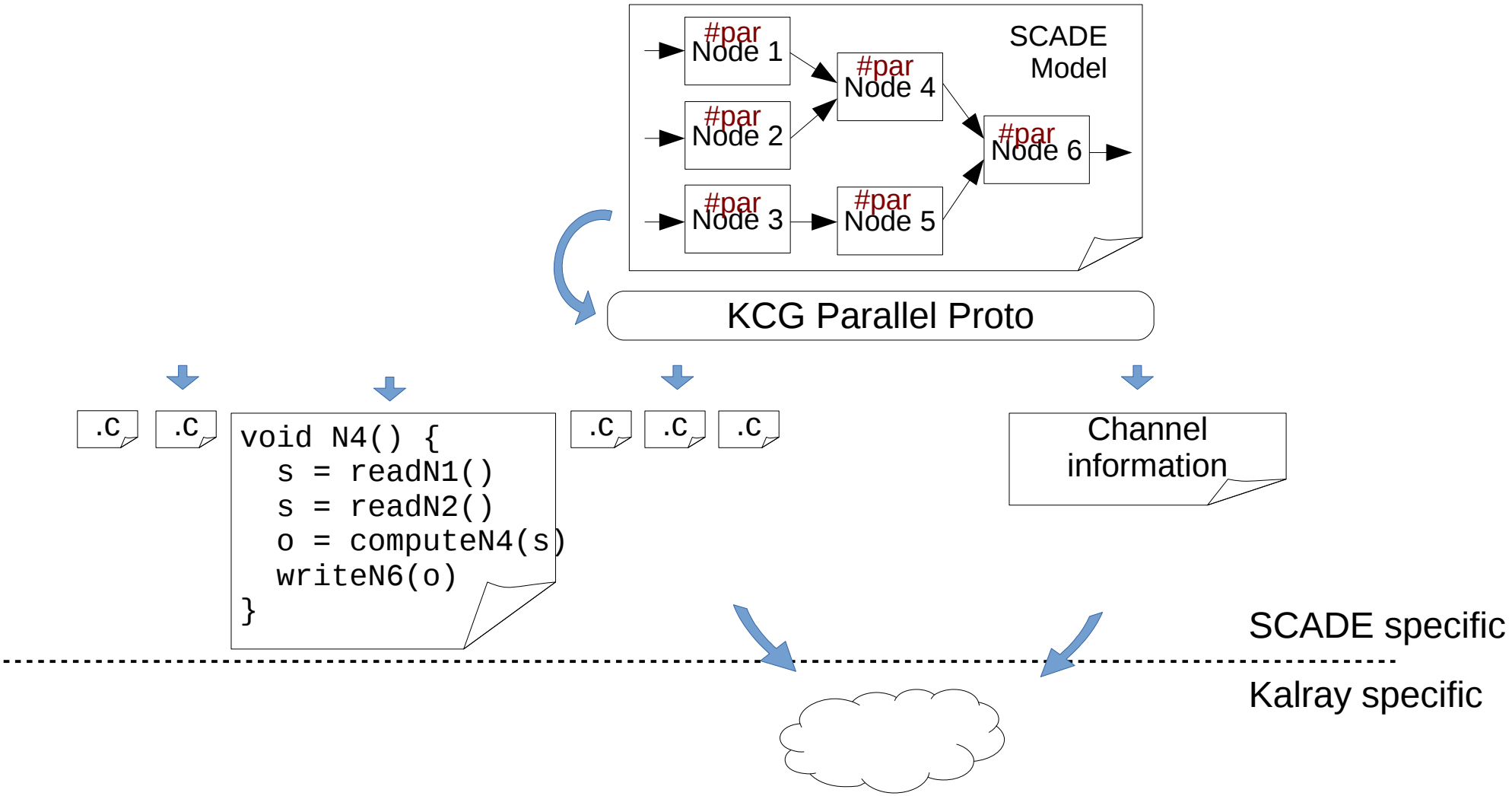


Issues

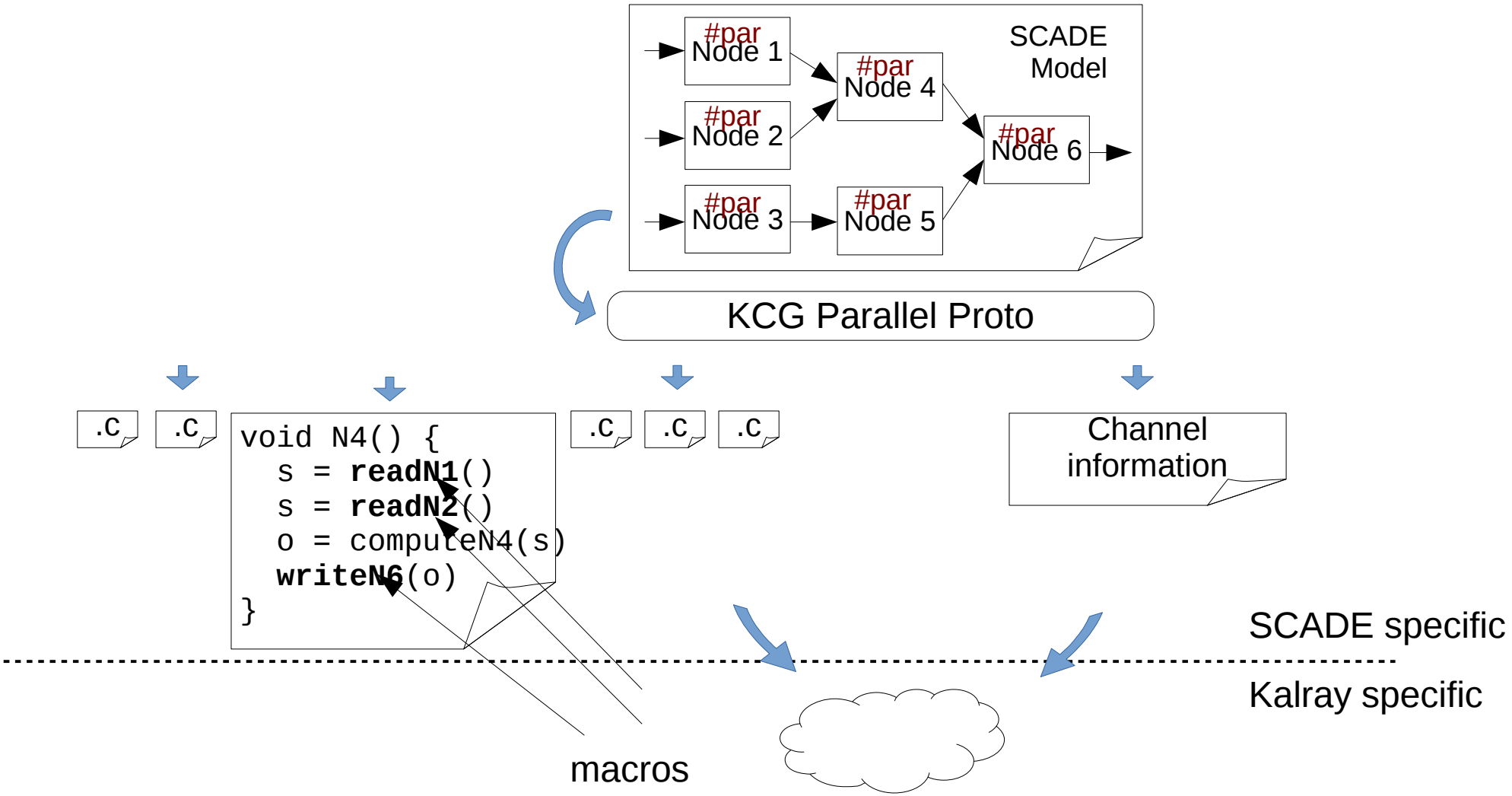


- Split into tasks
- Mapping
- Scheduling
- Communication

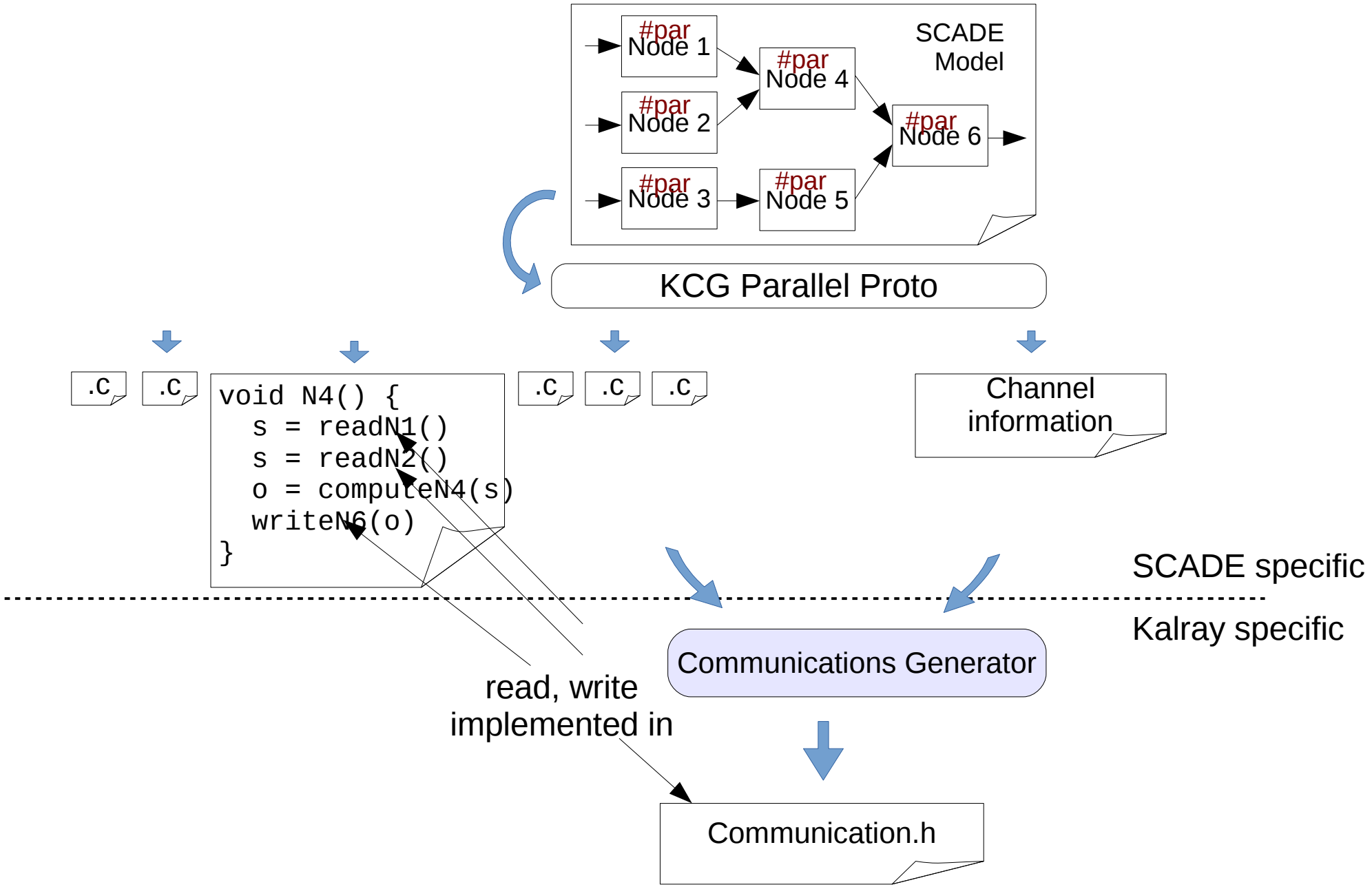
Communications



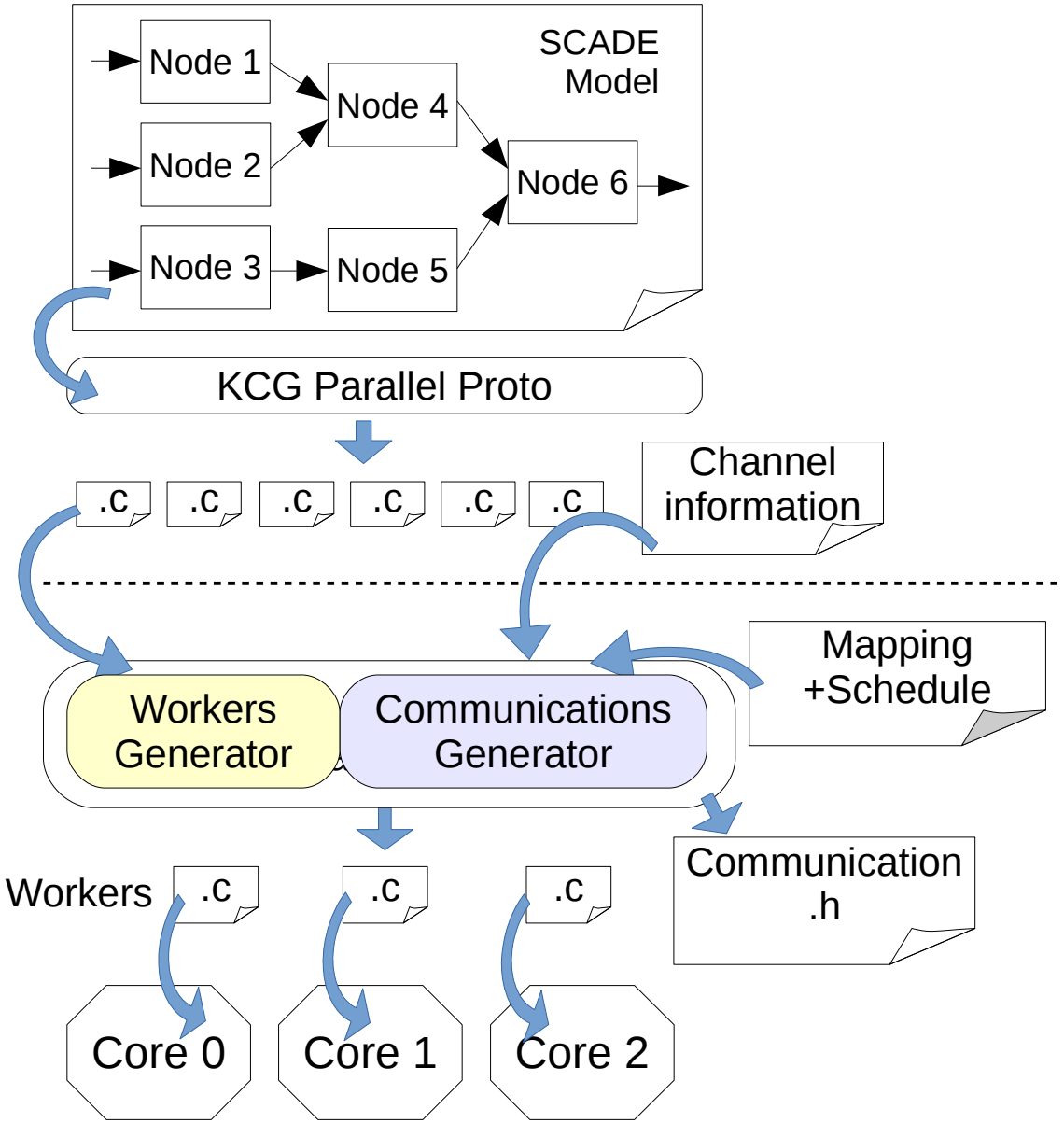
Communications



Communications



Contribution: Code Generator for Kalray MPPA



Kalray MPPA

- Generation from SCADE model
- Inputs:
  - Mapping (user)
  - Schedule (user)
  - Channel information (KCG)
  - Tasks (KCG)

Conclusion

- Code generation for MPPA
- Applied to ROSACE Flight Control case study (from Onera)
- Tool to bound shared-memory interferences: Hamza Rihani (Verimag)
- CAPACITES project: Kalray MPPA for critical applications.

Thank you!

Any Questions?