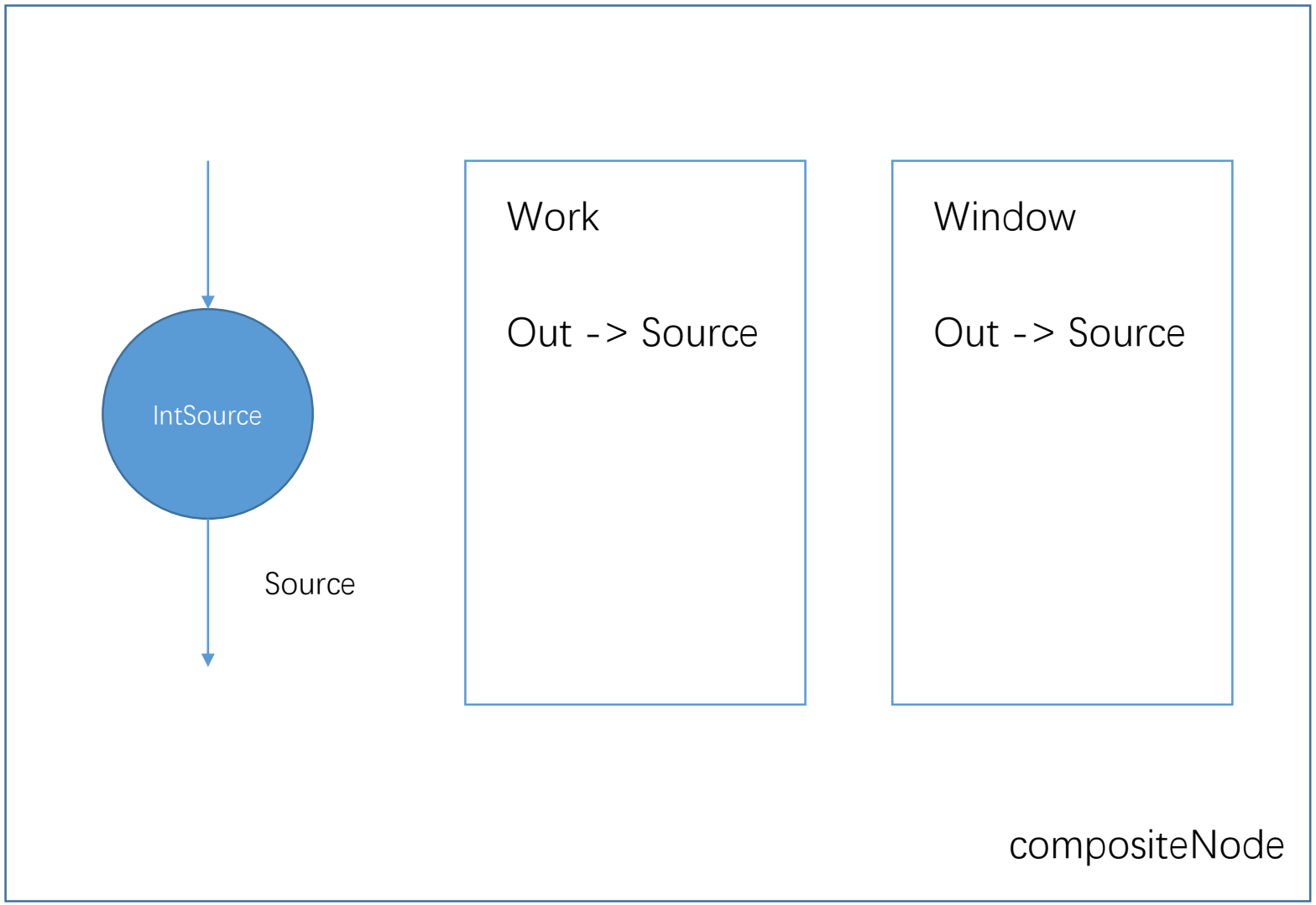
COStream后端学习分享

黄业兴 05-05

生成平面图 SSG

代码生成

改变调用的Composite的输入输出




```
composite Main ()
{
    stream<double x> Source,S1,S2,S3;
    int size = 8;
    Source = IntSource()();
    S1 = DCT_2D_Y(Source)(8);
    S2 = DCT_2D_X(S1)(8);
    S3 = DCT_Result(S2)();
    Sink(S3)();
}


composite IntSource(output stream<double x>
Out)
{
    Out = IntSource()
    {
        int i;
        init{
            i = 0;
        }
        work
        {
            if(i==64) i=0;
            //Out[0].x = From[i++];
            float x = pow(3, i++);
            int   y = (int) x;
            Out[0].x = y%75;
        }
        window{
            Out tumbling(1);
        }
    };
}
```

展开Composite节点

生成FlatNode

将Operator节点生成FlatNode

mapEdge2UpFlatNode
边和边的上端FlatNode

mapEdge2DownFlatNode
边与边的下端FlatNode

FlatNode

outFlatNodes
输 出 边各FlatNode

inFlatNodes
输 入 边各FlatNode

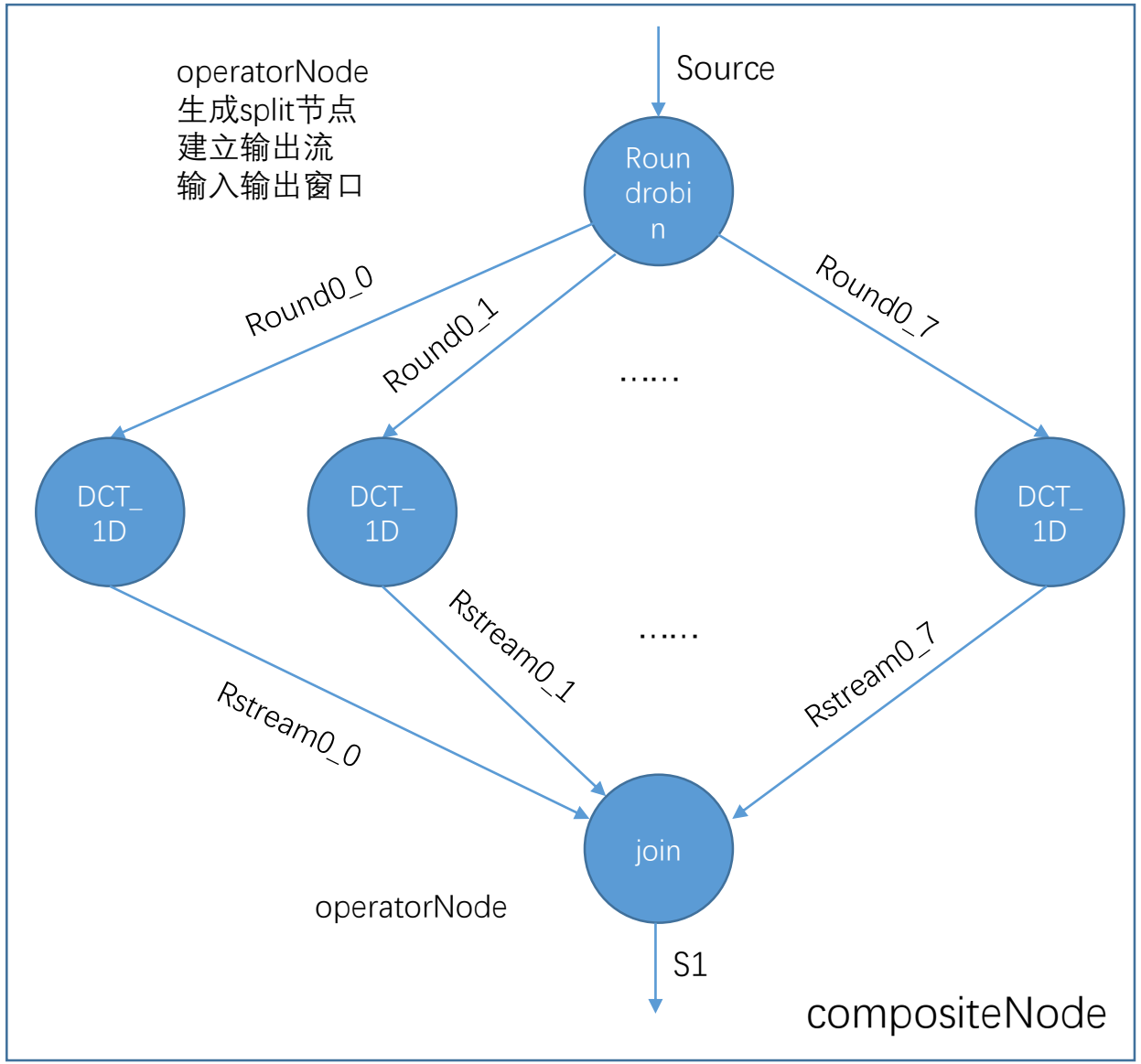
CompositeCall节点

CompositeNode节点

Operator节点

```
composite Main ()
{
    stream<double x> Source,S1,S2,S3;
    int size = 8;
    Source = IntSource()();
    S1 = DCT_2D_Y(Source)(8);
    S2 = DCT_2D_X(S1)(8);
    S3 = DCT_Result(S2)();
    Sink(S3)();
}

composite IntSource(output stream<double x>
Out)
{
    Out = IntSource()
    {
        int i;
        init{
            i = 0;
        }
        work
        {
            if(i==64) i=0;
            //Out[0].x = From[i++];
            float x = pow(3, i++);
            int   y = (int) x;
            Out[0].x = y%75;
        }
        window{
            Out tumbling(1);
        }
    };
}
```

展开Spilit Join节点



compositeCall节点
-> compositeNode
-> operatorNode

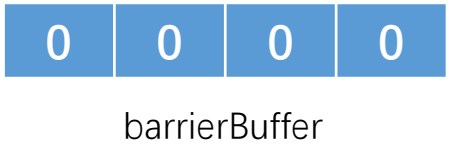
CompositCall

```
composite Main ()
{
    stream<double x> Source,S1,S2,S3;
    int size = 8;
    Source = IntSource()();
    S1 = DCT_2D_Y(Source)(8);
    S2 = DCT_2D_X(S1)(8);
    S3 = DCT_Result(S2)();
    Sink(S3)();
}
```

```
composite DCT_2D_Y(output stream<double
x>Out,input stream<double x>In)
{
    param
        int size;
    Out = splitjoin(In)
    {
        int i;
        split roundrobin(1);
        for (i = 0; i < 8; i++)
        {
            add DCT_1D(8);
        }
        join roundrobin(1);
    };
}
```

线程执行的控制

初始化



barrierBuffer


```
int i, sum;
do
{
    for (i = 1, sum = 1; i < n; i++)
        sum += barrierBuffer[i];
} while (sum < n); // 主线程等待其他线程执行完毕
for (i = 1; i < n; i++)
{
    barrierBuffer[i] = 0;
}
```

```
barrierBuffer[tid] = 1;
while (barrierBuffer[tid])
```

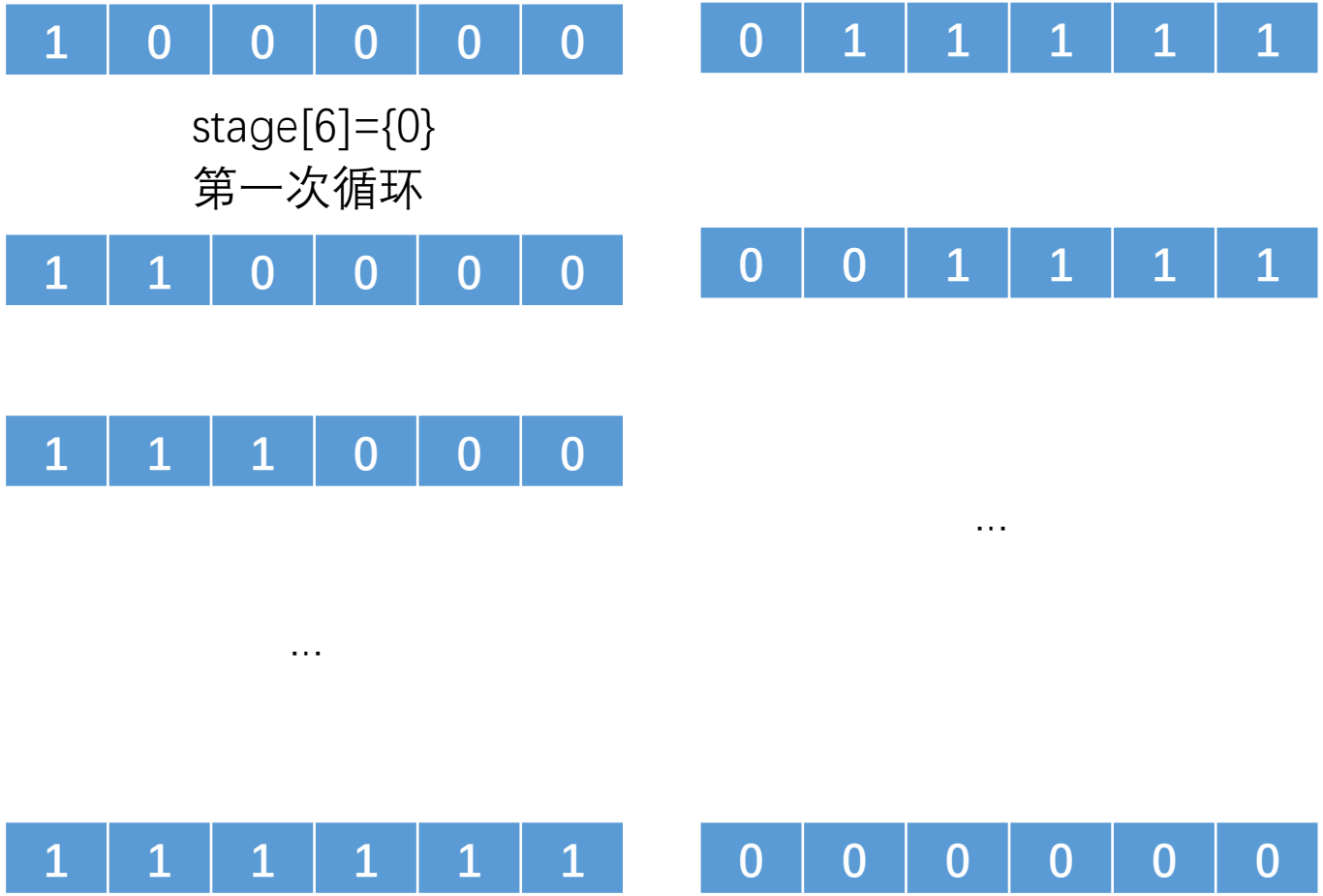
Stage = 0 的operator开始执行

```
int main(int argc,char **argv)
{
    void setRunIterCount(int,char**);
    setRunIterCount(argc,argv);
    set_cpu(0);

    allocBarrier(4);

    pthread_t tid[3];

    pthread_create (&tid[0], NULL,
thread_1_fun_start, (void*)NULL);

    pthread_create (&tid[1], NULL,
thread_2_fun_start, (void*)NULL);

    pthread_create (&tid[2], NULL,
thread_3_fun_start, (void*)NULL);

    thread_0_fun();

    return 0;
}
```

线程执行的控制

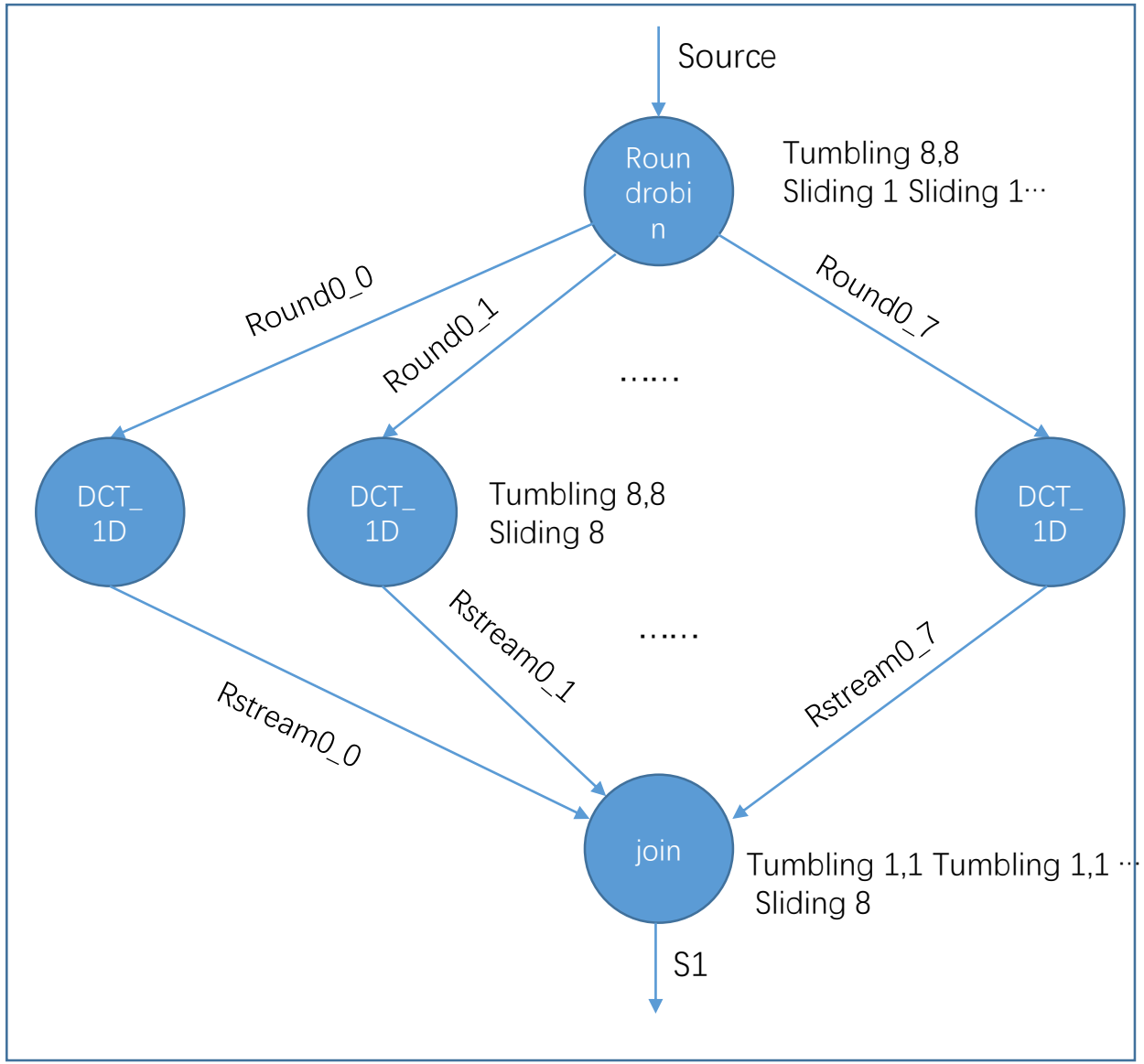
稳态调度

结束




```
for(int _stageNum=6;_stageNum<2*6+MAX_ITER-
1;_stageNum++)
    {
        if(stage[5])
        {
            Sink_22_obj.runSteadyScheduleWork();
        }
        for(int index=5; index>= 1; --index)
            stage[index] = stage[index-1];
        if(_stageNum == (MAX_ITER - 1 + 6))
        {
            stage[0]=0;
        }

        masterSync(4);
    }

for(int _stageNum=6;_stageNum<2*6+MAX_ITER-
1;_stageNum++){
        if(stage[0]){
         IntSource_0_obj.runSteadyScheduleWork();
Roundrobin_1_obj.runSteadyScheduleWork();
            DCT_1D_2_obj.runSteadyScheduleWork();
            DCT_1D_3_obj.runSteadyScheduleWork();
            DCT_1D_4_obj.runSteadyScheduleWork();
            DCT_1D_5_obj.runSteadyScheduleWork();
            DCT_1D_6_obj.runSteadyScheduleWork();
        }
        for(int index=5; index>= 1; --index)
            stage[index] = stage[index-1];
        if(_stageNum == (MAX_ITER - 1 + 6)){
            stage[0]=0;         }
        workerSync(1);
    }
```

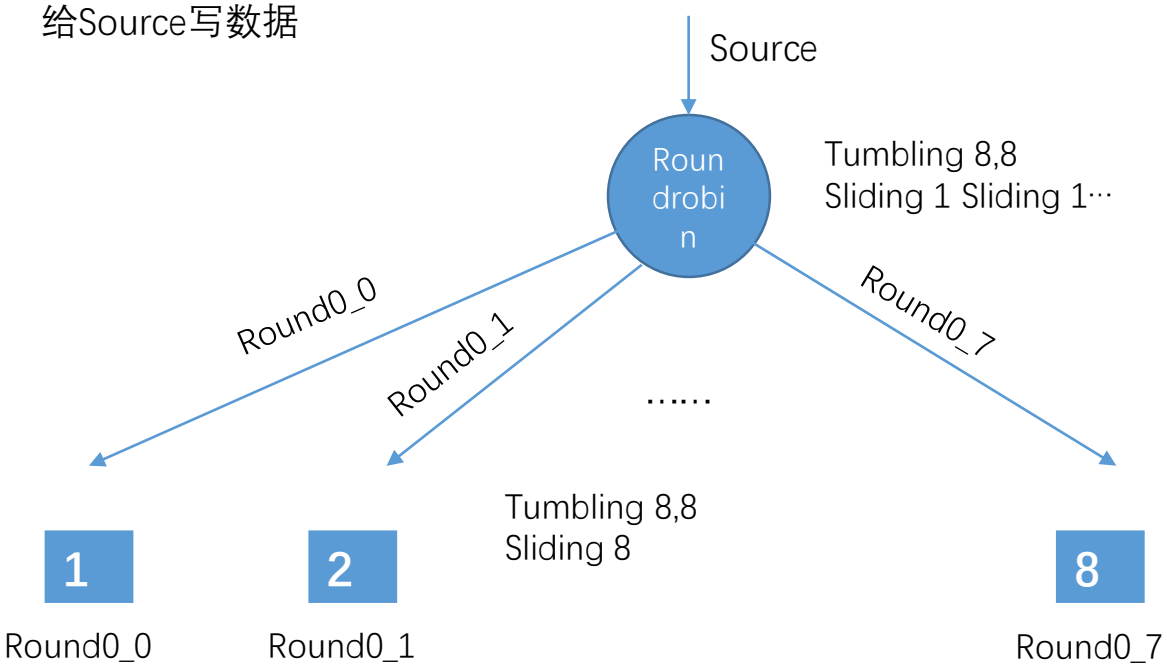
数据流的控制

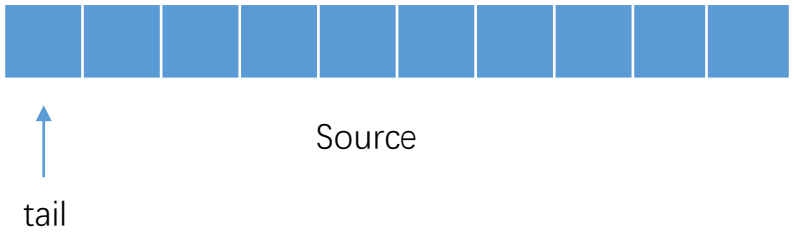



```c
int main(int argc,char **argv)
{
    void setRunIterCount(int,char**);
    setRunIterCount(argc,argv);
    set_cpu(0);

    allocBarrier(4);

    pthread_t tid[3];

    pthread_create (&tid[0], NULL,
thread_1_fun_start, (void*)NULL);

    pthread_create (&tid[1], NULL,
thread_2_fun_start, (void*)NULL);

    pthread_create (&tid[2], NULL,
thread_3_fun_start, (void*)NULL);

    thread_0_fun();

    return 0;
}
```

数据流的控制

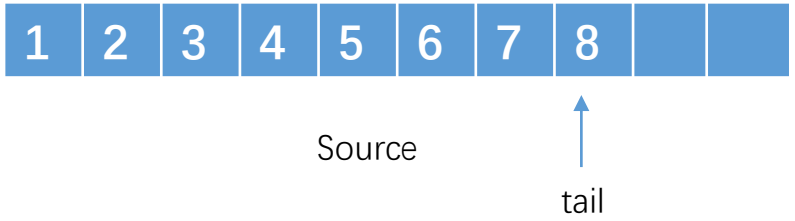
给Source写数据



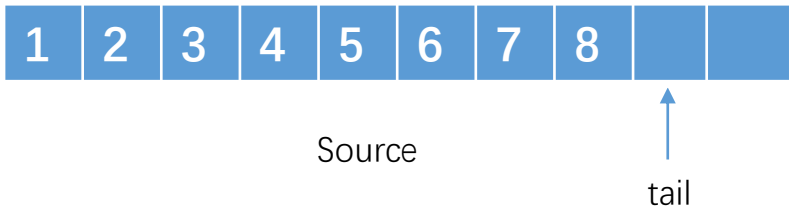



```
void work(){
   {
        if(i==64)   i=0;
        float x=pow(3,i++ );
        int y=(int)x;
        Source[0].x=y%75;
   }
```

```
Source.updateTail(1);
```

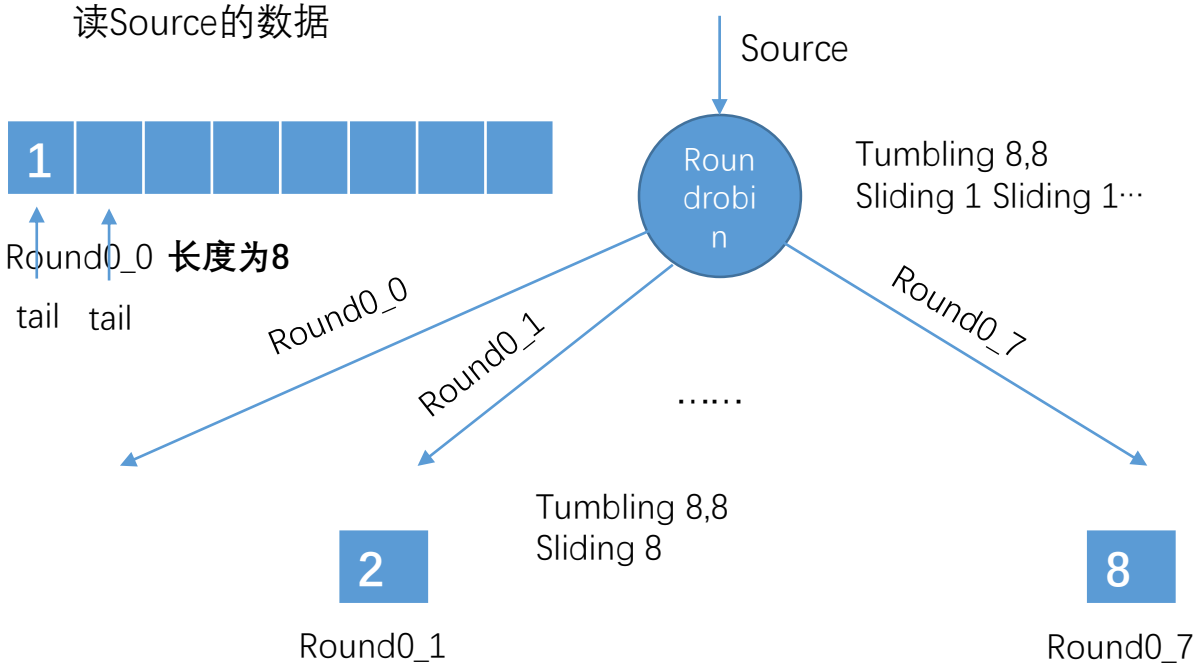



```
Source.resetTail();
```

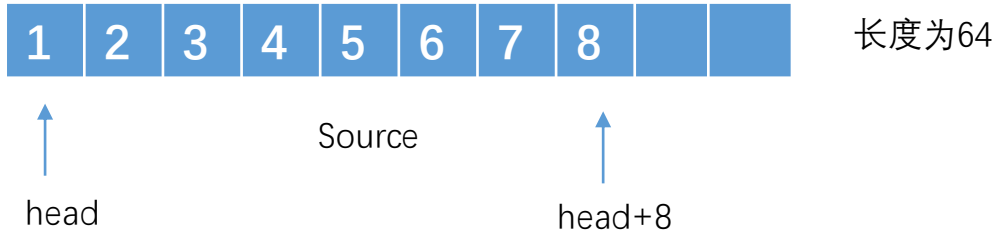


数据流的控制

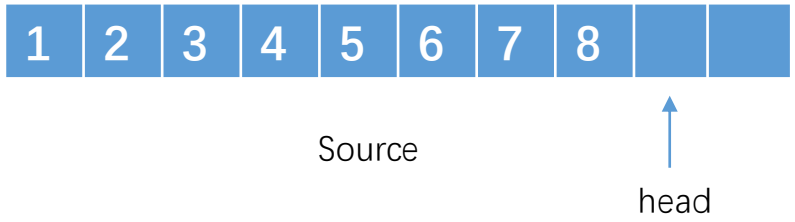
读Source的数据




```
Source.updatehead(8);
```




```
Source.resetHead();
```




```
for(i=0;i<1;++i)        round0_0[i]=Source[j++];
for(i=0;i<1;++i)        round0_1[i]=Source[j++];
for(i=0;i<1;++i)        round0_2[i]=Source[j++];
for(i=0;i<1;++i)        round0_3[i]=Source[j++];
for(i=0;i<1;++i)        round0_4[i]=Source[j++];
for(i=0;i<1;++i)        round0_5[i]=Source[j++];
for(i=0;i<1;++i)        round0_6[i]=Source[j++];
for(i=0;i<1;++i)        round0_7[i]=Source[j++];
```

```
round0_0.updatetail(1);
round0_1.updatetail(1);
round0_2.updatetail(1);
round0_3.updatetail(1);
round0_4.updatetail(1);
round0_5.updatetail(1);
round0_6.updatetail(1);
round0_7.updatetail(1);
```

```
round0_0.resetTail();
round0_1.resetTail();
round0_2.resetTail();
round0_3.resetTail();
round0_4.resetTail();
round0_5.resetTail();
round0_6.resetTail();
round0_7.resetTail();
```