Hackaton GENCI Yales2

Francesco GAVA – CORIA
Kévin Bioche - CORIA
Patrick BEGOU – LEGI
Ghislain LARTIGUE - CORIA

UMR 6614

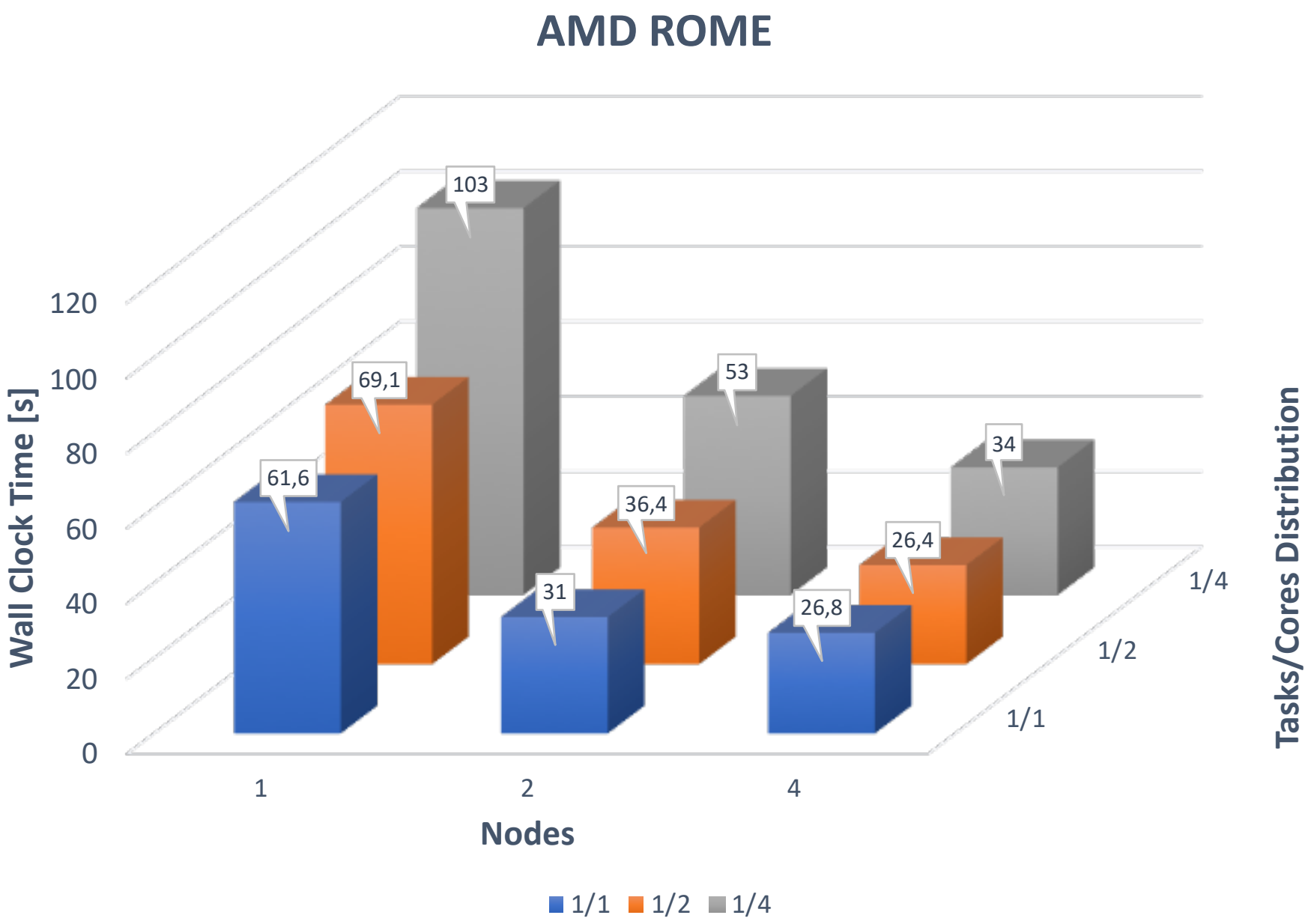
Objectives: AMD

- Compile
  - Yales2 ✓
  - Profiling tools ✗
- Performance measures
  - CPU-TIME as function of
    - Nelem/Core -> strong scaling ✓
    - Ncores -> weak scaling ✓
    - Nelem/group -> cache effect/Poisson (Not conclusive)
    - Graph (not fully implemented yet) ✗
    - Full vs partially empty nodes ✓
  - Memory
    - Bandwidth ✓
    - Cache misses ✗
    - …
  - Vectorization (maybe) ✗

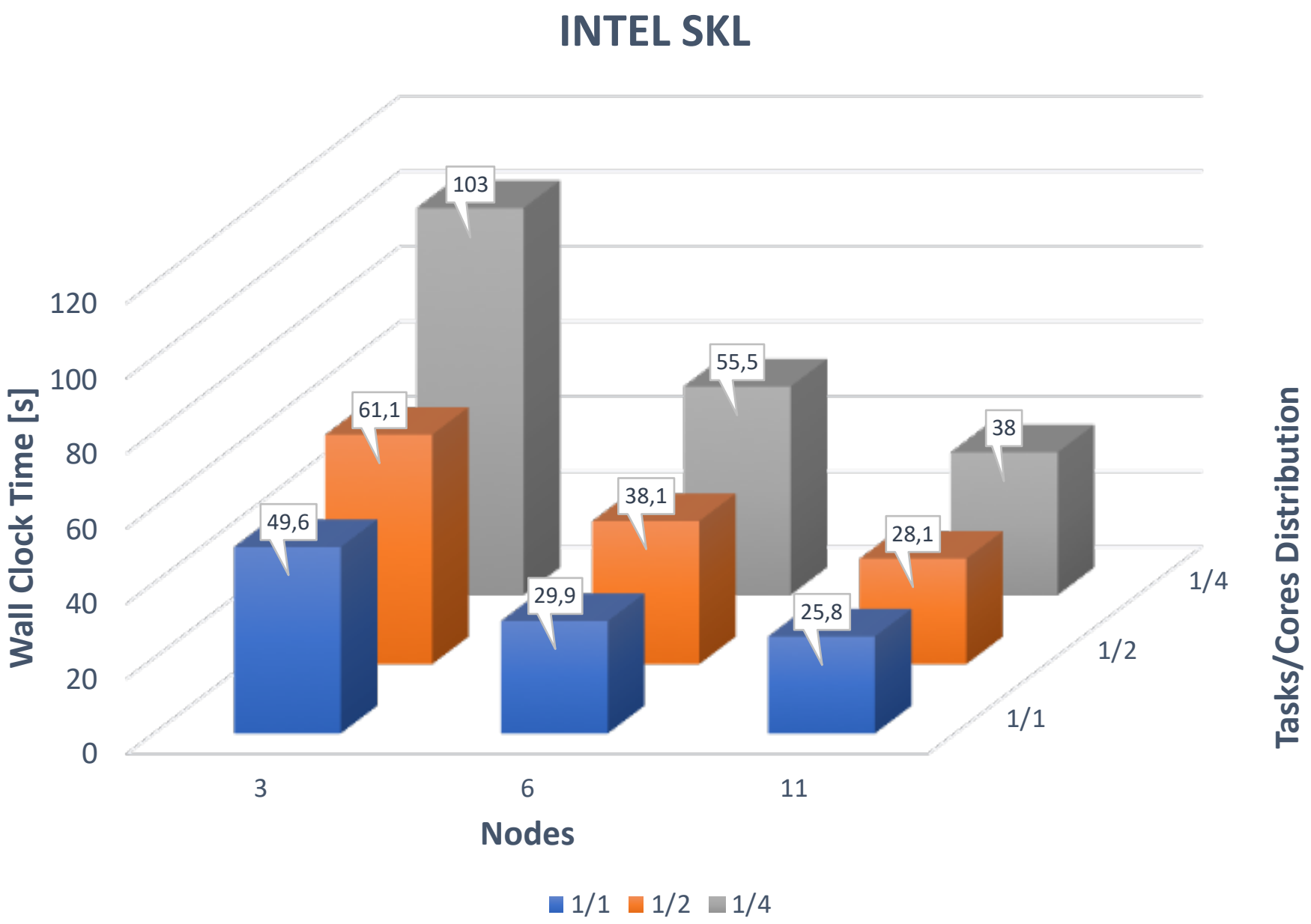
Objectives: AMD vs INTEL

- Comparison with INTEL
  - Theoretical architecture performances
    - FLOPS
      - LINPACK ✗
    - Memory Bandwidth
      - STREAM ✓
    - Network
      - MPI benchmarks ✗
  - YALES2 performances
    - Poisson ✓
    - Chemistry ✗
    - Diffusion ✗
    - …

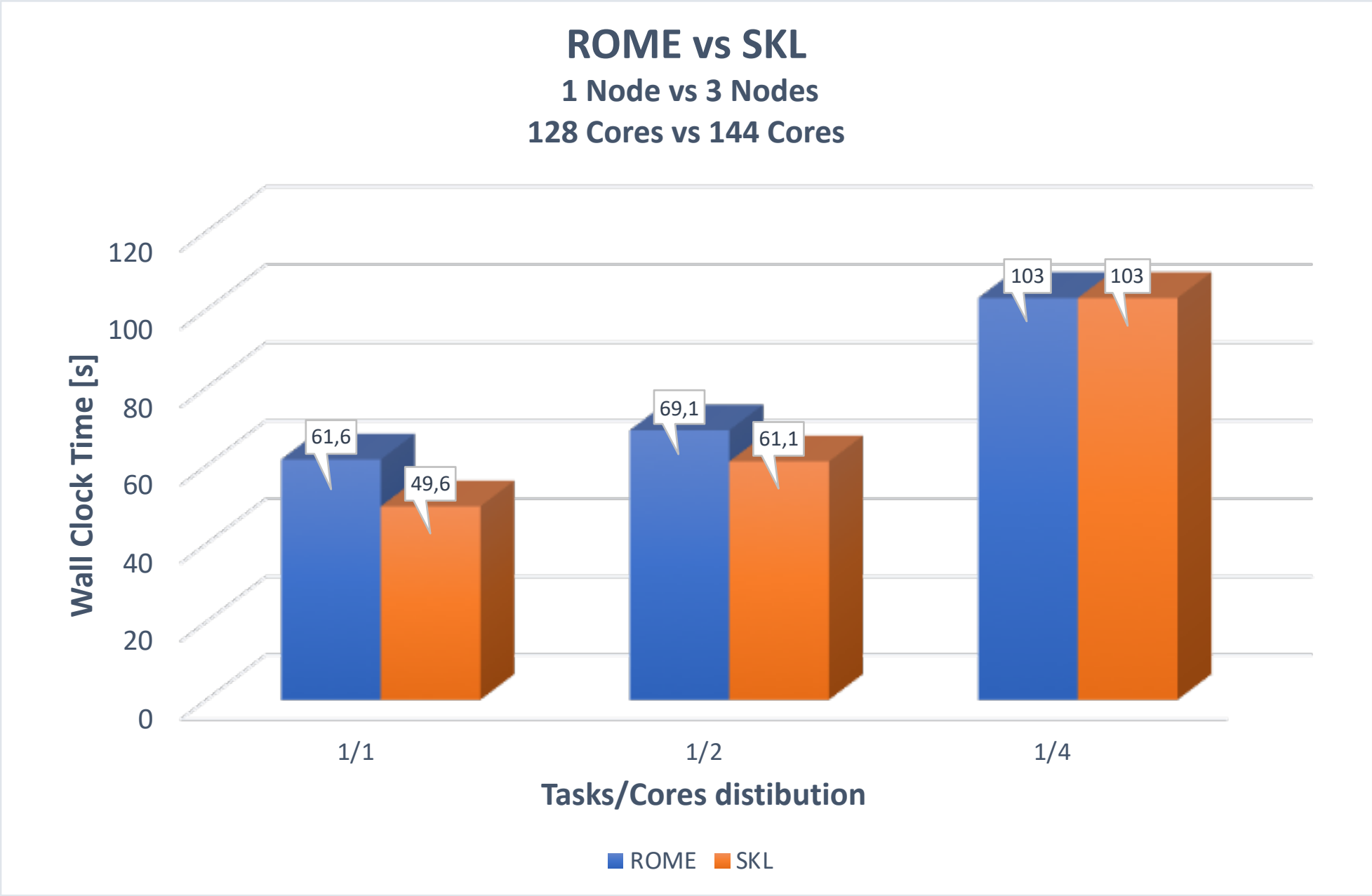
Data Analysis: AMD ROME



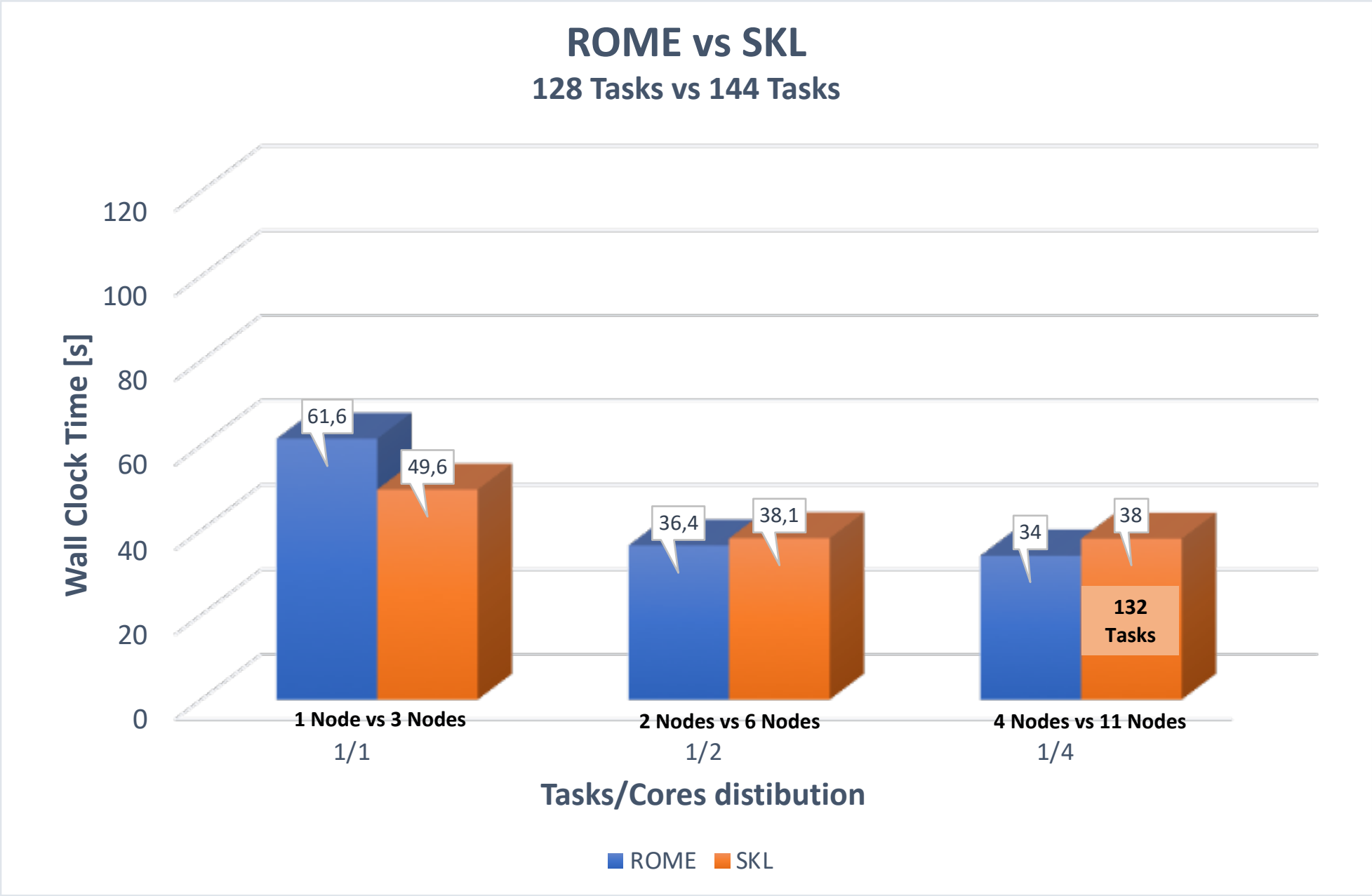
Data Analysis: INTEL SKL



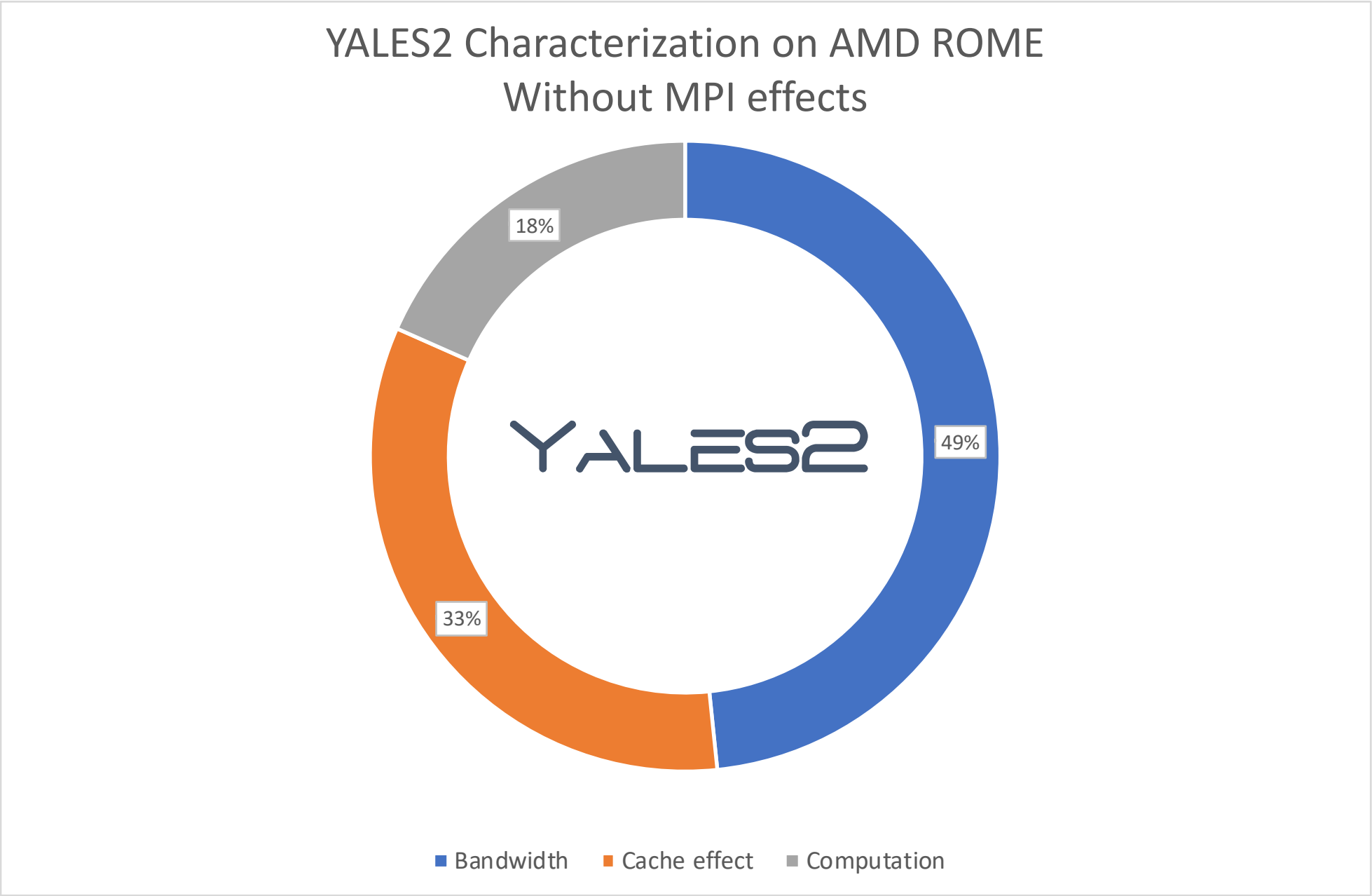
Data Analysis: AMD ROME vs IRENE SKYLAKE



Data Analysis: AMD ROME vs IRENE SKYLAKE



Data Analysis: AMD ROME vs IRENE SKYLAKE



Bonus: profiling tool in Yales2 by K. Bioche

- Activated with:

```
PROFILING = TRUE
NITER_SAMPLE_PROFILING = 1
```

- Dumps in a xml file:
  - Block with run characteristics
  - A block for each constant mesh with:
    - Mesh characteristics
    - Poisson's characteristics and evolution
    - Timers evolution
    - Timers statistics
  - Final global timers statistics
- Why?
  - Finer profiling timers on log
    - Only a little more intrusive
    - Avoid use in production anyway
  - Poisson's characteristics
    - Poissons's niter and deflation niter
  - Standardized output
    - To be coupled with a tool to exploit it
- To be continued:
  - A file for each worker (only master now)
  - Python tool for exploitation


```xml
<?xml version = '1.0' encoding='UTF-8' standalone='yes' ?>

<profiling>
  <run>
    <solver_name>2D_cylinder</solver_name>
    <date>
      <year>2020</year>
      <month>01</month>
      <day>31</day>
    </date>
    <time>
      <hour>01</hour>
      <minute>16</minute>
      <second>43</second>
      <millisecond>548</millisecond>
    </time>
    <n_workers>1</n_workers>
  </run>
  <!-- For a constant grid (between adaptations): grid, timers evolution and sta
  <constant_grid>
    <grid>
      <nelementpergroup>200</nelementpergroup>
      <nelem_global_total>24826</nelem_global_total>
      <nelem_global>24826</nelem_global>
      <nnode_global_total>12557</nnode_global_total>
      <nnode_global>12557</nnode_global>
      <nnode_global_total>12557</nnode_global_total>
    </grid>
    <poisson>
      <it_poisson_tab>[ 1 2 3 4 5 6 7 8 9 10]</it_poisson_tab>
      <niter_poisson_fine_tab>[ 88 86 74 67 62 76 63 63 54 49]</niter_poisson_f
      <niter_poisson_deflated_tab>[ 1772 1699 1447 1270 1071 1487 1104 1054 802
    </poisson>
    <timers>
      <name>TEMPORAL LOOP</name>
      <it_tab>[ 1 2 3 4 5 6 7 8 9 10]</it_tab>
      <time_tab>[  4.7594974260719196E-001  4.2276573973940640E-001  3.29702451
3258771685797716E-001  4.0608934121167523E-001  5.2663614773705766E-001  4.5702
      <rct_tab>[  3.7903141085226730E+001  6.7335468621391489E+001  3.938469989
339910825003789E+001  3.7729624757528692E+001  4.7931025175445247E+001  4.09456
      <averaged_time> 4.3562391798965647E-001</averaged_time>
      <averaged_rct> 4.3942881200095564E+001</averaged_rct>
      <std_dev_time> 5.6550209467726855E-002</std_dev_time>
      <std_dev_rct> 8.3607604196026308E+000</std_dev_rct>
```