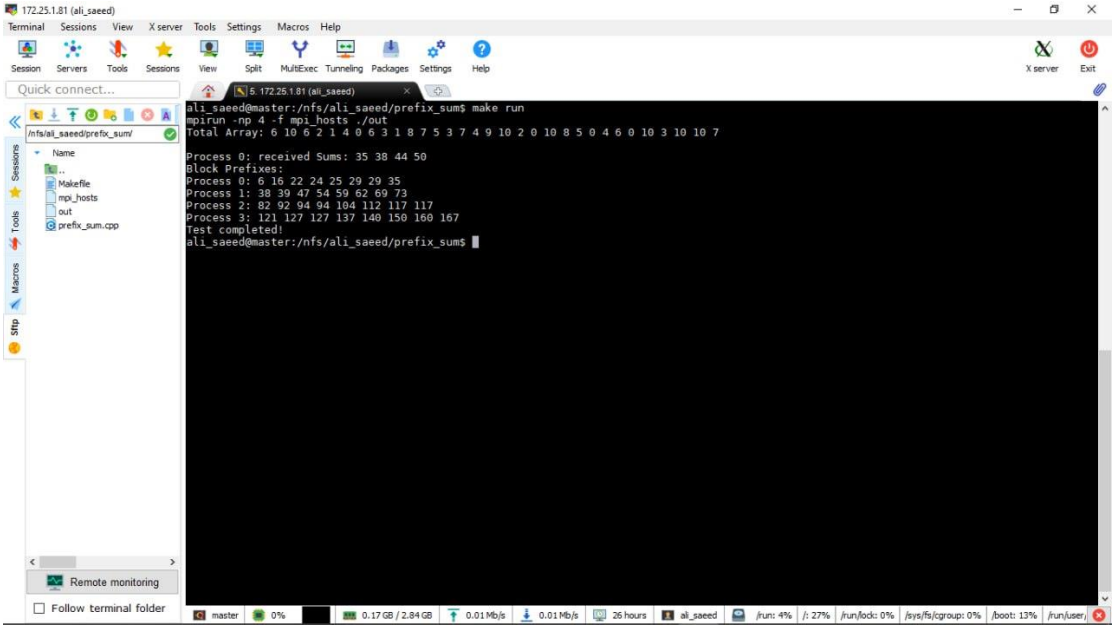
الحوسبة المتوازية

السؤال الأول: Prefix Sum

تم تجريب الكود على المثال التالي:

مصفوفة بطول 32 تحتوي قيم عشوائية من 0 إلى 10، وتم حساب مصفوفة مجموع البادئة على التوازي من قبل أربع اجرائيات وفق الخوارزمية المقترحة في نص السؤال، وكان الخرج على الشكل التالي:



يوضح الخرج مصفوفة مجموع البادئة التي قامت بحسابها كل إجرائية بعد تطبيق الخوارزمية المقترحة في نص السؤال، هذه المصفوفات يتم تجميعها في النهاية لدى الإجرائية ذات الرقم صفر ضمن مصفوفة لديها، وهي تمثل مجموع البادئة الكلي لمصفوفة الدخل.

السؤال الثاني: Reduction

ضمن طريقة tree reduce تشارك جميع العقد في القيام بالعمليات الحسابية، يتم بناء شجرة الجذر فيها هو الإجرائية ذات الرقم صفر، نفترض أن هذه الشجرة هي شجرة ثنائية كل إجرائية تستقبل مصفوفات من ابناءها وتضيفهم إلى مصفوفتها ومن ثمَّ ترسلهم إلى الإجرائية الأب.

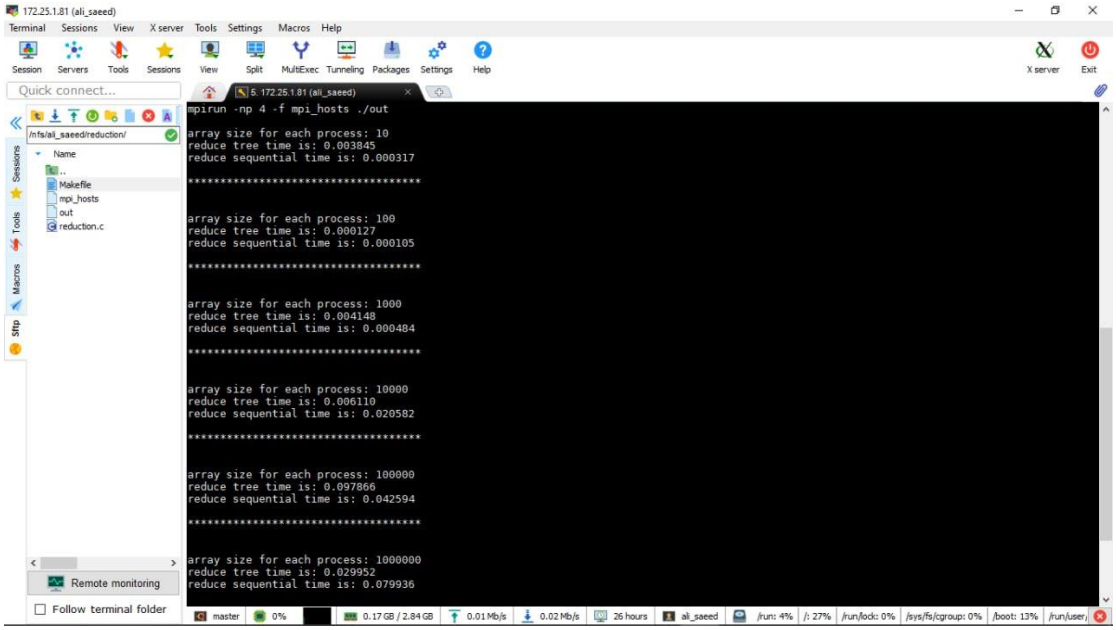
تعرف كل إجرائية من هم أبناءها ومن هو والده من خلال العلاقات:

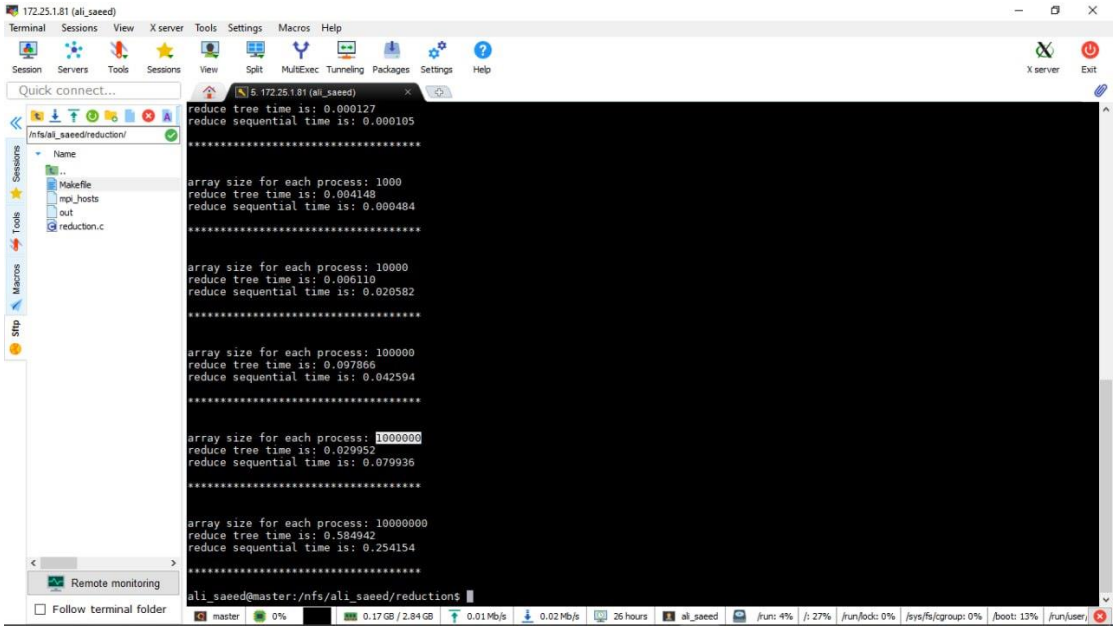
- Parent = (my_rank -1) / 2;
- Child1 = my_rank*2 +1;
- Child2 = my_rank*2 +2;

قد تكون الإجرائية ليس لديها أبناء لذلك يجب مقارنة رقم الأبن اذا ما كان أصغر من عدد الإجرائيات للتأكد بأن هذا الأبن موجود فعلاً.

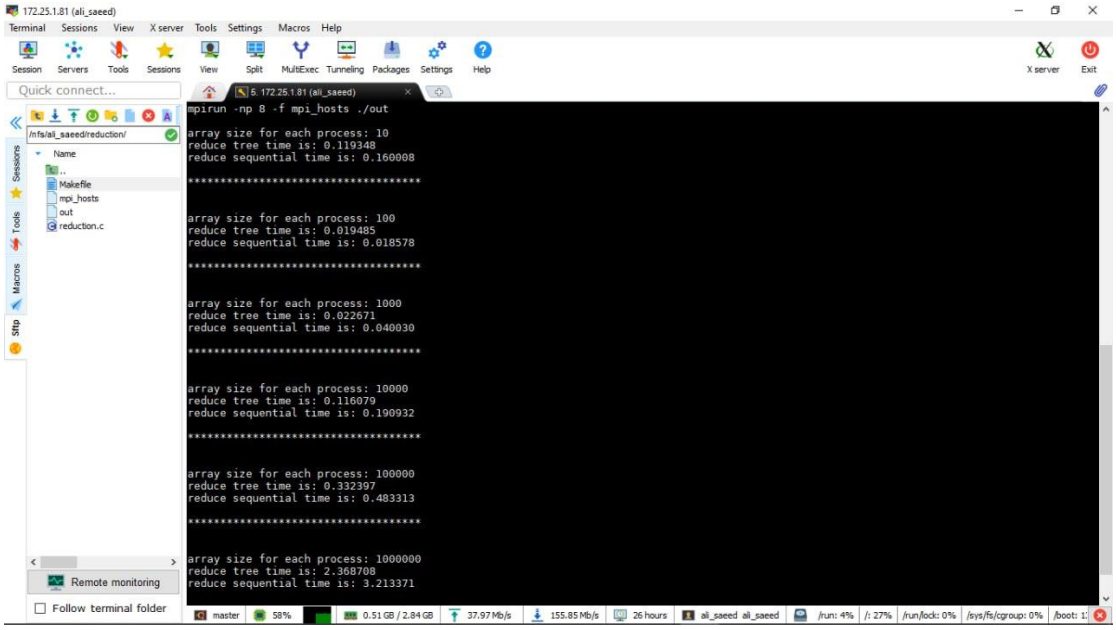
تم تجريب البرنامج من أجل أعداد مختلفة من الإجرائيات ومن أجل كل عدد من الإجرائيات تم تجريب أطوال مصفوفات مختلفة:

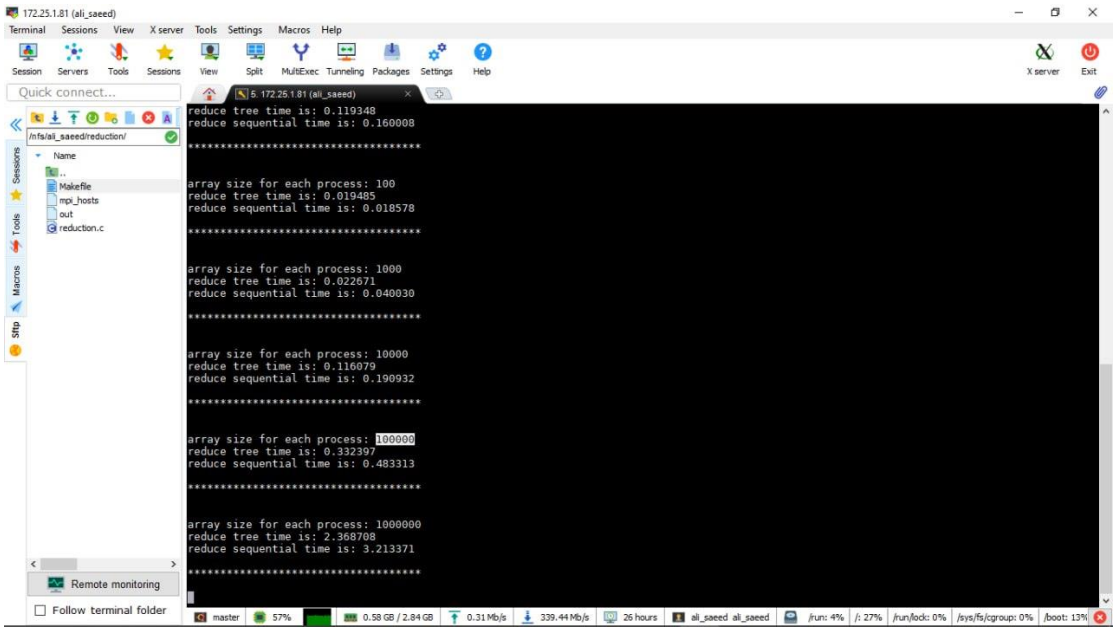
1) حالة 4 إجرائيات، كانت النتائج كما يلي





2) حالة 8 إجرائيات، كانت النتائج كما يلي





3) حالة 12 إجرائية، وهذه هو العدد الأعظمي من الإجرائيات التي يمكن أن تعمل معاً على التوازي ضمن العنقود الذي نعمل عليه، لأن العنقود يتألف من ثلاث عقد وكل منها تحتوي أربع معالجات منطقية، لذلك استخدام عدد أكبر من الإجرائيات يؤدي إلى جدولة الإجرائيات. كانت النتائج كما يلي

```
mpirun -np 12 -f mpi_hosts ./out

array size for each process: 10
reduce tree time is: 0.171733
reduce sequential time is: 0.341931

**********************************


array size for each process: 100
reduce tree time is: 0.095773
reduce sequential time is: 0.091694

**********************************


array size for each process: 1000
reduce tree time is: 0.058084
reduce sequential time is: 0.080056

**********************************


array size for each process: 10000
reduce tree time is: 0.104061
reduce sequential time is: 0.312437

**********************************


array size for each process: 100000
reduce tree time is: 0.792186
reduce sequential time is: 1.424273

**********************************


array size for each process: 1000000
reduce tree time is: 2.114793
reduce sequential time is: 8.051294
```

```
reduce tree time is: 0.095773
reduce sequential time is: 0.091694

**********************************


array size for each process: 1000
reduce tree time is: 0.058084
reduce sequential time is: 0.080056

**********************************


array size for each process: 10000
reduce tree time is: 0.104061
reduce sequential time is: 0.312437

**********************************


array size for each process: 100000
reduce tree time is: 0.792186
reduce sequential time is: 1.424273

**********************************


array size for each process: 1000000
reduce tree time is: 2.114793
reduce sequential time is: 8.051294

**********************************


array size for each process: 10000000
reduce tree time is: 27.280277
reduce sequential time is: 80.357920

**********************************

ali_saeed@master:/nfs/ali_saeed/reduction$
```

التعليق على النتائج:

من خلال النتائج السابقة يمكن أن نستنتج بأنه لا يمكن القول بأن أحد الطريقتين السابقتين أفضل من الأخرى بالمطلق، وإنما في بعض الحالات تكون طريقة tree reduce أفضل من sequential reduce وتتعلق هذه الحالات بشكل عام بعدة عوامل:

- عدد الاجرائيات: حيث أن زيادة عدد الإجرائيات تلعب دوراً كبيراً في تخفيض زمن الحساب، ومن جهة أخرى زيادة عدد الإجرائيات يؤدي إلى زيادة زمن التواصل حيث أن عمليات التواصل بين الإجرائيات ستزداد.
- طول المصفوفة: كلما كان طول المصفوفة كبير هذا سيؤدي إلى جعل عملية التواصل أكثر صعوبة، وسيزيد من زمن الحساب.

لذلك عند مقارنة الطريقتين sequential reduce وtree reduce يجب أن نأخذ بعين الإعتبار العاملين السابقين (عدد الإجرائيات وطول المصفوفة)، وهنا نناقش عدة حالات:

1. **عدد الإجرائيات كبير وطول المصفوفة صغير:**

- طريقة reduce tree: في هذه الحالة ولكون عدد الإجرائيات كبير سيكون عدد مرات التواصل بين الإجرائيات كبير ولكن في نفس الوقت لا يتطلب وقتاً كثيراً لأن طول المصفوفة قليل، بالنسبة لزمن الحساب سيكون صغير جداً لأنه أساساً الحسابات عددها قليل بالإضافة إلى أنه تم تقسيمها على عدد كبير من الإجرائيات وبالتالي سيكون زمن الحساب صغير جداً.
- طريقة sequential tree: في هذه الحالة سيكون زمن الحساب قليل لأن عدد العمليات الحسابية المراد تطبيقها قليل. في مثل هذه الحالة يمكن أن نقول أن أداء طريقة tree reduce يتقارب من أداء طريقة sequential reduce. حيث أنه في مثل هذه الحالة على الرغم من أنه في طريقة reduce tree سيكون زمن الحساب صغير جداً ولكن عملية التواصل أيضاً ستتطلب وقت بسبب عدد الإجرائيات الكبير.

2. **عدد الإجرائيات كبير وطول المصفوفة كبير:**

طريقة tree reduce تتفوق بشكل واضح من حيث سرعة الأداء على طريقة sequential reduce، لأن الحسابات ستكون كثيرة جداً وتتطلب وقت كبير لذلك تقسيمها على الإجرائيات سيقلل زمن الحساب كبير، على الرغم من أن زمن التواصل لن يكون قليل في هذه الحالة ولكن مع ذلك تقسيم الحسابات على الإجرائيات سيعطي أداء أفضل بشكل واضح أكثر من أن تقوم إجرائية واحدة (الإجرائية صفر) بكامل الحسابات لوحدها. (زمن الحساب يهيمن على زمن الاتصال والانتظار).

3. **عدد الإجرائيات صغير وطول المصفوفة صغير:**

في مثل هذه الحالات يمكن القول بأن أداء طريقة sequential reduce يتفوق على أداء طريقة tree reduce بشكل بسيط جداً لأنه في هذه الحالة زمن الحساب صغير ولا يتطلب وقت كبير، لذلك وجود زمن التواصل بين الإجرائيات التي لن تقلل وقت الحساب بشكل كبير (لأنه أساساً صغير) لن يكون ذات جدوى كبيرة لذلك من الأفضل استخدام sequential reduce.

4. **عدد الإجرائيات صغير وطول المصفوفة كبير:**

في مثل هذه الحالة يمكن القول أنه بشكل عام يكون أداء طريقة sequential reduce أفضل من أداء طريقة tree reduce، لأن زمن الحساب في هذه الحالة كبير جداً ولكون عدد الإجرائيات صغير لن يتم تخفيض بشكل كبير وبالمقابل سيكون التواصل بين الإجرائيات يتطلب وقت كبير لأن طول المصفوفة كبير (عدد مرات التواصل قليل ولكن تتطلب وقت كبير) لذلك من الأفضل استخدام sequential reduce.

5. في جميع الحالات السابقة يجب الانتباه إلى أن طريقة tree reduce لا تُنفذ على التوازي بشكل كامل أي أنه في لحظة معينة لا تكون جميع الإجرائيات تعمل معاً، وإنما كما ذكرنا سابقاً ستكون الإجرائيات الآباء تنتظر الأبناء حتى تنتهي من مهامها وترسل مصفوفاتها إلى الآباء، وزمن الانتظار يجب أن يتم أخذه بعين الإعتبار في طريقة reduce tree.

6. نلاحظ في الحالات التي يكون فيها طول المصفوفة كبير، في طريقة reduce tree ستساهم الإجرائيات (العقد الداخلية) في تنجيز الحسابات، وكما ذكرنا في البند 5 في هذه الأثناء سيكون الآباء ينتظرون الأبناء حتى ينتهو من حساباتهم وبالتالي زمن انتظار أطول، لذلك كما ذكرنا في الحالات التي يكون فيها طول المصفوفة كبير، في حال كان عدد الإجرائيات كبير فإن هذا الأمر سيقلل من زمن الحساب كثيراً بما يغطي على الوقت الضائع في التواصل والانتظار، أما عندما يكون عدد الإجرائيات قليل فإن على الرغم من تقليل زمن الحساب ولكن لن يكون كافياً للهيمنة على زمن الانتظار والاتصال.
7. نلاحظ أن التواصل في طريقة tree reduce سيتم عبر عدة مراحل:
المرحلة الأولى: تقوم كل إجرائية باستلام مصفوفة بطول N من كل ابن من أبناءها، كل إجرائية لديها ابنين على الأكثر أي عدد الأبناء ثابت، أي تقوم كل إجرائية بإجراء عملية استلام مرتين على الأكثر لذلك:

Tcomm1 = O(2N) = O(N)

المرحلة الثانية: ستقوم كل إجرائية بعملية ارسال واحدة على الأكثر تتضمن مصفوفة بطول N إلى الأب، لذلك:

Tcomm2 = O(N)

الحساب والتواصل لا يتم بشكل تفرعي كامل، وانما عبر عدة مراحل من التوازي والتسلسل والانتظار.

8. زمن الحساب **التسلسلي** الكلي سيكون من رتبة O(N*P).

الطالب: علي حسان سعيد.