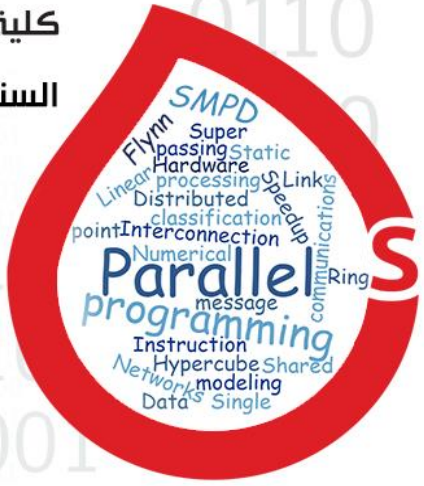
Mpi & Pvm



د. روان قرعوني

29/04/2024

RB Informatics;

بسم الله الرحمن الرحيم

تعرفنا في المحاضرة السابقة على:

- قانون Amdahl للاستجابة العظمى
- Static process creation
- Dynamic process creation
- نوعي التنفيذ المتزامن و الغير متزامن
- Group message

سنتعرف في هذه المحاضرة على أهم التوابع المستخدمة في القسم العملي، للتعرف على أساسيات البرامج التي سيتم إعطائها لاحقا بإذن الله .

مقدمة:

يُعد كل ابن عبارة عن ملف exe يتخاطب مع ملفات exe أخرى -تمثل الأبناء- عن طريق pvm

تتم عملية التخاطب بواسطة توابع الإرسال والاستقبال

تعرفنا في القسم العملي على التوابع الأساسية للإرسال والاستقبال:

تابع الإرسال pvm-sendوتتم عملية الإرسال على 3 مراحل

1. pvm-init (تهيئة)
2. pvm-pkint (تحزيم)
3. pvm-send (الإرسال)

يمكن تشبيه عملية إرسال الرّسالة بتهيئة صندوق (التهيئة)، ثم وضع عدد معين من الأغراض فيه (التحزيم)، ثم إرسال الصندوق كاملا (الإرسال).

يمكننا تحزيم عدة أنواع مختلفة من البيانات في نفس الرسالة (int, String, float,.....)

لكل نوع من أنواع البيانات تابع التحزيم الخاص

pvm_pkint()

pvm_pkfloat()

pvm_pkstr()

تذكر: يجب مقابلة كل تابع sendبتابع recv لضمان عدم توقف البرنامج.

تابع الاستقبال Pvm-recv تتم عملية الاستقبال أيضا ل 3 عمليات:

- pvm_recv() (استقبال)
- pvm_bufinfo() (قراءة معلومات ال بافر)
- pvm_upkint() (فك التحزيم)

نعود إلى مثال إرسال الصندوق:

في عملية استقبال الرسالة، تشبه المرحلة الأولى (الاستقبال)استلام الصندوق المُرسل، ثم تليه المرحلة الثانية (قراءة معلومات ال buffer) وهي تشبه قراءة المعلومات الموجودة على غلاف الصندوق لمعرفة تفاصيله لتمييز الصندوق الهدف ,وأخيرا فك التحزيم للحصول على المعلومات المطلوبة .

ملاحظة:

يستخدم التابع pvm_bufinfo() لتمييز الرسالة المُرادة من بين عدة رسائل موجودة , فيمكن تجاهل استخدامه في حال التأكد من استلام الرسالة الصحيحة . مثلا في حالة عدم وجود رسائل أُخرى.

هكذا نكون قد أمنا عملية التخاطب بين ملفات ال exe

هناك ما ستَعَلّمُك إيّاه طول التجربة؛

أنَّكَ إذا تفَرّغتَ تتعَب..

كيف تتم عملية إنشاء الأبناء في pvm؟

عند تشغيل ال pvm (عندما نقوم بتشغيل البرنامج) ، أول ما يعمل هو الأب (parent) , في حال حاجة الأب لملفات exe (أبناء) يقوم باستدعاء التابع pvm_spawn() لهذا التابع العديد من البارامترات، سنتعرف على البارامتر الأساسي، وهو مصفوفة عناصرها هي id كل ابن.

لكل مهمة task id يميزها عن غيرها.

سؤال ما هو خرج التابع pvm_spawn ؟ ما هي قيمة المتغير x ؟ int x = pvm_spawn ()

هو عدد الأبناء التي تم إنشائها بنجاح، حيث عند استدعاء التابع نقوم بتحديد عدد الأبناء المُراد إنشائها لكن قد لا يتمكن من إنشاء العدد المطلوب لأسباب مثل امتلاء الذاكرة أو حدوث خطأ معين ...

سؤال ما الفائدة من معرفة عدد الأبناء التي تم إنشائها بنجاح؟

ليكن لدينا برنامج يقوم بضرب مصفوفتين ويحتاج إلى إنشاء 10 أبناء للقيام بهذه العملية في حال عدم إنشاء العدد المطلوب هذا سيؤدي إلى حدوث أخطاء في النتيجة.

ما الحل؟

الحل الأول

هو التحقق من عدد الأبناء المُنشئة قبل المباشرة بحل المسألة

```
numtids = pvm_spawn(taskname, argv, 0, "", 10, tids);

    if (numtids == 10) {
        printf("Good , you can continue\n");
    } else {
        fprintf(stderr, "Error: Not enough children were created.\n");
        pvm_exit();
        return 1;
    }
```

الحل الثاني

هو الاكتفاء بالأبناء المُنشئة ففي مثال ضرب مصفوفتين باستخدام 10 أبناء , لنفرض مثلا تم إنشاء 8 أبناء في هذه الحالة يمكن استخدام هذه الأبناء و تأدية عمل الأبناء التي لم تنشأ بواسطة الأب أو إرسالها إلى أحد الأبناء الثمانية المُنشئة.

سنتعرف على مثال لكلا الحالين في القسم العملي.

pvm_mytid()
قيمة خرجه هي ال task id الخاص بل ابن الذي يستدعي التابع .

pvm_parent_tid()
قيمة خرجه هي ال task idالخاص ب الأب للابن الذي استدعى التابع .

Group in pvm

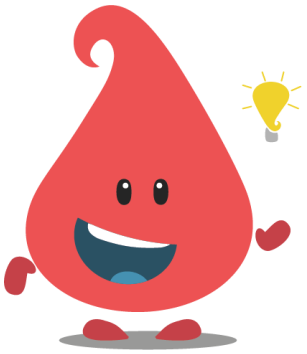
تذكر:
ال group هو مجموعة من المهام المحددة.

- في ال pvm لا يوجد group افتراضي هذا يعني أن مهمة خلق الغروب تكون مسؤولية المبرمج
  تتم عملية خلق غروب جديد من خلال التابع pvm_joingroup ("ABC")
  كما تستخدم التعليمة نفسها لإضافة مهمة جديدة إلى غروب.

- عند خلق المهام الجديدة باستخدام spawn تعمل كل مهمة على حدى وأول مهمة تقوم بتنفيذ التابع السابق تقوم بخلق ال group وتقوم باقي المهام بالانضمام إلى نفس المجموعة .

توضيح: في حال كان لدي 10 مهام قمنا باستدعاء التابع Pvm-joingroup("ABC") في 6 مهام منها أول مهمة تقوم بتنفيذ التابع تقوم بخلق الغروب و تقوم المهام الخمسة المتبقية بالانضمام لنفس المجموعة و تأخذ كل مهمة رقم خاص بها ضمن المجموعة

تذكر : الفائدة من استخدام ال group هي تهيئة الرسالة المُراد إرسالها لمرة واحدة فقط .



تكلمنا في المحاضرة السابقة عن التوابع المسؤولة عن إرسال الرسائل في groupبشكل نظري

سنكمل اليوم بشرح إضافي عنهم .

1. pvm-gather("ABC")
2. pvm-scatter("ABC")
3. pvm-reduce("ABC")
4. pvm- bcast ("ABC")

لكل مجموعة يوجد مهمة هي ال root

عند استخدام التوابع السابقة يجب أن نقوم بتحديد هذه المهمة

توضيح تحدثنا في المحاضرة السابقة أن التابع gather يقوم بتجميع الرسائل الواردة من عدة مهام لكن كيف نميز بين المهمة التي تقوم بالتجميع وباقي المهام؟

يتم ذلك من خلال تحديد المهمة ال root في بارامترات التوابع السابقة.

pvm_mcast()

- تعرفنا في المحاضرة السابقة على طريقة ال multicast يستخدم هذه التابع لتحديد ids المهام التي سوف يتم إرسال الرسالة إليها
- ملاحظة: عند استخدام ال multicast لإرسال الرسالة لعدة مهام لا يتم إنشاء مجموعة خاصة بهذه المهام بل يقوم بإرسال الرسالة حسب مصفوفة ال ids

الفروق بين ال mpi و pvm

pvm	mpi
تعد ال pvmمكتبة جاهزة	بينما mpiتعد interface
لذلك نستطيع استخدام mpiبدون الحاجة لوجود virtual machine	
بيمنا في pvm تكون عملية إنشاء الأبناء عند الحاجة أثناء التنفيذ.	في ال mpi عند بداية تنفيذ التطبيق تظهر واجهة لتحديد عدد الأبناء اللازمة لكل مهمة .
- في pvmعند بداية التنفيذ تكون المهام المُنشئة مستقلة (لا يوجد مجموعة افتراضية تجمع بينهم)	في ال mpi عند بداية التنفيذ جميع المهام المُنشئة تصبح ضمن مجموعة أساسية تدعى MPI_COMM_WORLD"(محيط الاتصال العالمي)

ملاحظة: في ال mpi يصبح اسم المجموعة communicator"" بدلا من "group"

هل هذا يعني أنه لا يمكن إنشاء مجموعات في mpi ؟

نستطيع أن نقوم بإنشاء communicator يضم عدد معين من المهام ضمن MPI_COMM_WORLD

أي مجموعة فرعية ضمن المجموعة الأساسية MPI_COMM_WORLD

ملاحظة:

في mpi لم يعد يمكن وصف المهام في المجموعة ب "أبناء" بل "مهام" حصرا , لأن جميع المهام متساوية .

- mpi-comm-size

  خرجه عدد المهام في ال communicator

- mpi-comm-rank

  يعيد ترتيب المهمة ضمن ال communicator

  يكون ترتيب أول مهمة أُنشأت هو الصفر (0) أي ان قيم "ترتيب المهمة" تبدأ من الصفر.

ما الفائدة من "ترتيب المهمة"؟

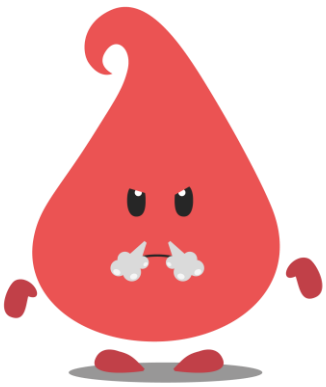
القدرة على التخاطب بين المهام كما نستخدمه لتحديد المهمة الـ root نلاحظ أن ال rank في ال mpi مشابه لعمل ال id في ال pvm

- MPI_Init()

  يبدأ البرنامج دائما به في حال تجاهل استخدامه فإن جميع توابع ال mpi لن تعمل

- MPI_Finalize()

  ينتهي البرنامج دائما به.



توابع الإرسال والاستقبال في حالة التنفيذ المتزامن

- MPI_Send
- MPI_Recv

توابع الإرسال والاستقبال في حالة التنفيذ غير المتزامن

- MPI_Isend
- MPI_Irecv

- MPI_Waitall

لها العديد من البارمترات أهمها هي مصفوفة من ال requests تحدد طلبات الإرسال التي يجب انتظارها.

إذن ما الفائدة من استخدام هذا التابع؟

ضمان استلام القيمة المُراد استخدامها

فمثلا أريد تغيير قيمة المتغير x=8 لضمان الحصول على النتائج الصحيحة يجب أولا أن أتأكد من استلام قيمة المتغير

ما الفائدة في حالة التنفيذ غير المتزامن مع استخدام التابع MPI_Waitall؟

ممكن أن نقوم بتنفيذ كود معين لا يحتاج إلى القيمة غير المُستلمة قبل الوصول إلى أول تعليمة تحتاج هذه القيمة.

توابع ال communicator

- MPI_Gather
- MPI_Scatter
- MPI_Reduce
- MPI_Bcast

ملاحظة:

لدينا التابع mpi_Barrier يجب استخدامه بعد استخدام توابع ال communicator

في حال الحاجة لضمان استلام البيانات قبل إرسال بيانات أُخرى تعتمد على وجود البينات الأولى .

مثال للتوضيح

لنفرض أننا قمنا بعملية إرسال باستخدام التابع MPI_Scatter أريد إرسال بيانات أُخرى بعد ضمان إتمام عملية الإرسال في هذه الحالة يجب استخدام التابع MPI_Barmer .

لاحظ الشبه بين

التابع MPI_Barmer في توابع الcommunicator

والتابع MPI_Waitall في الإرسال غير المتزامن

MPI_Reduce_scatter

ليكن لدي 4 مهام

كل مهمة تريد إرسال مصفوفة بطول 4 عناصر إلى المهام (المهام المتبقية ونفسها)

نقوم بتطبيق scatterكما تعلمنا في المحاضرة السابقة

عندها سيتم إرسال القيم عند ال index = 0 في كل من المصفوفات الأربعة للمهمة الأولى

والقيم عند ال index =1في المصفوفات الأربعة للمهمة الثانية

والقيم عند ال index =2في المصفوفات الأربعة للمهمة الثالثة

والقيم عند ال index =3في المصفوفات الأربعة للمهمة الرابعة

هكذا نقوم قد طبقنا الإرسال باستخدام scatter

بعد ان تستلم كل مهمة القيم المُرسلة لها تقوم بتطبيق reduce **عليها , أي تقوم بتنفيذ عملية معينة على القيم المُستلمة .**

ليكن لدينا المثال التالي:

T1	T2	T3	T4
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

بعد تطبيق reduce **وبفرض كانت العملية هي الجمع**

ينتج

T1	T2	T3	T4
0	4	8	12

سؤال عند تنفيذ ال reduce هل أستطيع تنفيذ عملية مختلفة في كل مهمة ؟

لا يجب أن تقوم جميع المهام بتنفيذ نفس العملية (جمع , or , ...and)

- MPI_AlltoAll()

ليكن لدينا 4 مهام وكل مهمة تريد إرسال مصفوفة بطول 4 عناصر للمهام (المهام الأخرى و المهمة نفسها) كما في التابع السابق تقوم الهمام أولا بعملية إرسال باستخدام scatter , بعد تنفيذ الإرسال يصبح لدي في كل مهمة مصفوفة جديدة .

مثال:

T1	T2	T3	T4
0	5	9	13
1	6	10	14
2	7	11	15
3	8	12	16

1	2	3	؟
5	6	7	8
9	10	10	12
12	15	10	16

T1	T2	T3	T4
1	2	3	4

يمكن تلخيص عمله بإنه يقوم بقلب السطر إلى عامود و العامود إلى سطر

نلاحظ أن التابع MPI_AlltoAll يقوم بنفس عمل التابع MPI_Reduce_scatter لكن بدون تنفيذ أي عملية على الرسائل المُستلمة

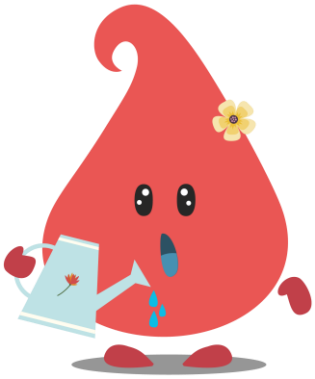
- MPI_Scan ()

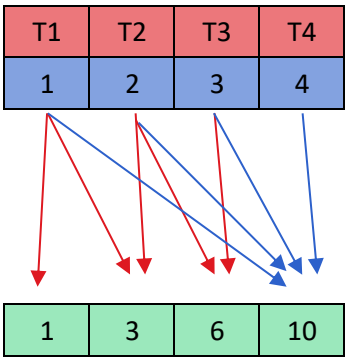
ليكن لدينا 4 مهام وكل مهمة تملك قيمة معينة تقوم كل مهمة بالقيام بعملية -تحدد ضمن بارمترات التابع- على القيمة الموجودة لديها و القيم الموجودة لدى المهام التي قبلها !!

للتوضيح

تقوم المهمة الأولى بتنفيذ العملية على القيمة التي لديها فقط لأنه لا يوجد مهام قبلها

بينما تقوم المهمة الثانية بتنفيذ العملية على قيمتها وعلى القيمة الموجودة في المهمة الأولى (قبل التعديل) أما المهمة الثالثة تقوم بتنفيذ العملية على كل من قيمتها وقيم المهمة الأولى والثانية





تعد التوابع الثلاث السابقة توابع تجميعية متقدمة.

سؤال ما الفائدة من هذه التوابع التجميعية المتقدمة؟

سنجد حالات معينة تقوم هذه التوابع بتوفير الوقت اللازم لكتابة أكثر من أمر إرسال كما يوجد العديد من التوابع الأُخرى المشابهة.

$$t_p = t_{comm} + t_{comp}$$

زمن تنفيذ التفرعي = زمن التواصل + زمن المعالجة

تكلمنا سابقا عن أهمية زمن التواصل وأنه غير مهمل ويعد السبب بعدم الحصول على النتائج المُثلى بينما زمن الحساب فهو موجود في حالة البرمجة التسلسلية أو البرمجة التفرعية.

نستنتج أنه يجب أن نقوم بتقليل زمن التواصل قدر الممكن.

يعد زمن التواصل ناتج جمع كل من زمن إرسال قيمة مضروب بعدد القيم المُراد إرسالها وزمن التهيئة (التحزيم والإرسال..)

$$t_{comm} = t_{startup} + n * t_{data}$$

لدينا النسبة $\frac{t_{comp}}{t_{comm}}$

أي مقسوم زمن الحساب على زمن التواصل تعد هذه النسبة معيار لجودة البرنامج تكون الجودة أقل عند اقتراب هذه النسبة من الواحد أي يكون زمن التواصل مساوي لزمن المعالجة

كيف أستطيع تقليل زمن التواصل؟

من خلال اختيار طريقة الربط الأنسب حسب المسألة تعرفنا على طرق الربط في المحاضرة الثانية

The End...

تأمَّل حياتك جيّدًا، ضع جانبًا كلّ ما قتل وقتك، وأخذ شيئًا من صحّتك، وهَدَم بُنيانًا طالما حَلُمتَ به، لا مزيد من الفراغ، لا مزيد من العجز، المُمكِن الّذي بينَ يديك، أحَقّ أن تُعطيه كُلّك..