مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (36) العدد (3) 2014
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (36) No. (3) 2014

تصميم خوارزمية نقل للملفات بأزمنة متوازنة اعتماداً على المسارات المتعددة للمعالجة

الدكتورة مريم ساعي*
الدكتور جبر حنا**

(تاريخ الإيداع 2 / 10 / 2013. قُبِل للنشر في 19/ 5 / 2014)

□ ملخّص □

يقدم هذا البحث خوارزميةً جديدةً لنقل الملفات ونسخها ضمن نظام التشغيل بزمن متوازن بغض النظر عن حجم الملفات باستخدام المسارات المتعددة للمعالجات، وذلك بتقسيم الملفات إلى مجموعات جزئية، بحيث تكون متساوية في الحجم قدر الإمكان، ويتم نقلها على التوازي مما يؤمن إنقاصاً ملحوظاً في زمن عملية النقل، ويزداد مقدار الإنقاص للزمن مع تزايد الحجم الكلي للملفات المنقولة. كما يعرض هذا البحث عملية تنفيذ هذه الخوارزمية من خلال لغة البرمجة C#.net ويقدم بعض الاختبارات المنجزة للتحقق من أداء الخوارزمية وموثوقيتها.

الكلمات المفتاحية: مسارات المعالجة –نقل الملفات– المشغولية

* مدرس– قسم هندسة الحاسبات والتحكم الآلي – كلية الهندسة الميكانيكية والكهربائية – جامعة تشرين – اللاذقية– سورية.
** أستاذ – قسم هندسة الحاسبات والتحكم الآلي – كلية الهندسة الميكانيكية والكهربائية – جامعة تشرين – اللاذقية– سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (36) العدد (3) 2014
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (36) No. (3) 2014

Balanced File Transfer Time Algorithm Design Based on Multithreading

Dr. Mariam M. Saii*
Dr. Jabr Hanna**

(Received 2 / 10 / 2013. Accepted 19 / 5 / 2014)

□ ABSTRACT □

A new algorithm for copying and transferring file in OS with balanced transfer time using multithreading is proposed.The new algorithm depends on segmenting file to a number of group with balanced size, that way reduces the transfer time, the total file increase the reduced time increase.

This algorithm is implemented by C#.net and many result obtained by the test which clearly show the readability of this algorithm

Keywords: multithreading – file transfer – delay

* **Assistant Professor, Computer and Control Engineering Department, Faculty of Mechanical and Electrical Engineering , Tishreen University, Lattakia, Syria.**

** **Professor, Computer and Control Engineering Department, Faculty of Mechanical and Electrical Engineering , Tishreen University, Lattakia, Syria.**

مقدمة:

تعد عمليات نقل الملفات بشكل عام الجزء الأكبر من الأعمال التي تقوم بها وحدات المعالجة، وهذه العمليات تتوزع على المستويات كافة سواء كانت ضمن القرص الصلب أو بين القرص الصلب وقرص قابل للإزالة أو من عمليات تناقل الملفات عبر الشبكة.

تعددت الأبحاث حول عمليات نقل الملفات والأزمنة المتعلقة بها واستخدمت العديد من الآليات لإنقاص أزمنة النقل وتحسين أداء تطبيقات الزمن الحقيقي، فقد قام الباحث Damien Le Moal مع مجموعة من الباحثين [1] بتصميم نظام نقل خاص بملفات الفيديو والصوت وأطلقوا عليه اسم AVFS بحيث يحقق التوافق الأمثل بين مساري الدخل والخرج من جهة ومسارات التخزين على القرص الصلب من جهة ثانية، كما توجه البعض الآخر من الباحثين مثل Huan Wang و[2]Wen Song للاهتمام بنقل الملفات عبر الشبكة بكفاءة عالية واستخدموا شبكات Petri من أجل تحسين عمليات النقل, ولا تزال عمليات التطوير في آليات النقل موضع الاهتمام والبحث فعمليات نقل الملفات ضمن نظام التشغيل لا تزال تعاني من تأخيرات ملحوظة بالرغم من التطور الحاصل للمعالجات وأنظمة التشغيل.

تؤثر عمليات نقل الملفات بشكل أو بآخر على عمل التطبيقات الأخرى وفي بعض الأحيان تأخذ فترات زمنية لا بأس بها، مما يضطر المستخدم للتوقف عن العمل ريثما تنتهي عملية النقل أو النسخ أو غيرها، فعمليات النسخ أو النقل تستهلك قسماً من أداء وحدة المعالجة المركزية بحيث تبقي مجالاً لعمل التطبيقات الأخرى، وتستمر لفترة زمنية ملحوظة ولكن القسم المخصص يؤثر بشكل ملحوظ على التطبيقات الأخرى قيد التشغيل لذلك كانت فكرة هذا البحث هي زيادة القسم المخصص لعميات النقل بهدف إنقاص زمن عملية النقل بشكل ملحوظ ، وهذا ما يمكن تحقيقه من بناء مسارات معالجة متعددة.

إن استخدام المسارات المتعددة يعني توزيع الملفات المنقولة إلى عدة أجزاء أو قوائم ملفات File Lists ليتم نقل كل قائمة أو عدة قوائم من خلال إحدى هذه المسارات، ولا يتم تنفيذ هذه العملية بالشكل الأمثل إلا من تحقيق توزيع متوازن للملفات بين هذه القوائم، وبقدر ما تكون أحجام هذه القوائم متقاربة تكون نهاية عمليات النقل ضمن المسارات متقاربة وبالتالي يصبح الزمن الكلي لعملية النقل هو الزمن اللازم لنقل الملفات ضمن أحد المسارات فقط وليس الحجم الكلي، وبالتالي هناك معامل إنقاص Reducing Factor للزمن تتناسب مع عدد الملفات المتواجدة ضمن القائمة الواحدة، لذلك يمكن تحقيق أمثلية عمل لمثل هذه العمليات بتحديد حجم مناسب لكل قائمة بحيث يكون زمن النقل لهذا الحجم مناسب نسبياً للمستخدم ولا يشكل زمناً ملحوظاً ، ومن ثم تصميم خوارزمية توزيع للملفات المراد نقلها على مجموعة قوائم ملفات ذات حجم أعظمي هو الحجم الأنسب للنقل ككتلة واحدة، ومن ثم توزيع هذه القوائم على مجموعة مسارات معالجة ليتولى كل مسار عملية النقل الخاصة.

تحتاج عملية تحقيق مثل هذه الخوارزميات إلى بيئةً برمجيةً تمتاز بقدرة جيدة من حيث التعامل مع الأغراض بالإضافة إلى التعامل الجيد مع مسارات المعالجة من حيث البناء والتخاطب والهدم.

أهمية البحث وأهدافه:

يعدُّ هذا البحث ذا أهمية كبيرة لجميع مطوري البرمجيات وأنظمة التشغيل سواء كانت أنظمة تشغيل عامة كأنظمة ....Windows, Unix أو أنظمة تشغيل خاصة لبعض أنظمة التحكم عن بعد أو التجهيزات التي تتطلب نظام للتشغيل وتحتاج في عملها لنقل ملفات.

تتلخصالأهداف التي يحققها البحث بالآتي:

- تطوير عمليات نقل الملفات بالاعتماد على المسارات المتعددة.
- إنقاص الزمن اللازم لعمليات نقل الملفات.
- تحقيق زمن نقل أعظمي للملفات بغض النظر عن الحجم.
- زيادة استقلالية عمليات النقل عن حجم الملفات.
- تحقيق خوارزمية نقل الملفات كدعم لنظام التشغيل.

طرائق البحث ومواده:

1 التعريف بمسارات المعالجة[5,6,13]:

المسار هو عبارة عن تدفق متسلسل وحيد للتحكم، فكل البرامج تحوي مساراً واحداً على الأقل وهو المسار الرئيس ويمكن أن تنفذ أكثر من مسار واحد في نفس الوقت (التوازي الوهمي، أي يبدو كأنهما ينفذان بنفس الوقت).

من الواضح أن المسارات مفيدة من أجل تحسين سرعة الحساب على المعالجات المتعددة، ولكن من أجل الحواسيب ذات المعالج الوحيد، فإن الحسابات التي تتم عن طريق المسارات يمكن أن تتم بدونها.

2 الحاجة للمسارات المتعددة:

كل مسار له مكوناته الخاصة وهي: عداد البرنامج ومكدس التحكم (ينادي ويعيد الإجرائيات) ومكدس البيانات (المتحولات المحلية local variables)[7,8,13].

ولكل المسارات مكونات مشتركة وهي كود البرنامج والمراكم والصنف ومتحولات الحالة.

المسار كما هو معروف خفيف الوزن بالمقارنة مع العملية لأنه يستخدم الموارد الأساسية أي الذاكرة ونظام التشغيل بشكل قليل، كما أنه سريع الإنشاء والتلف.

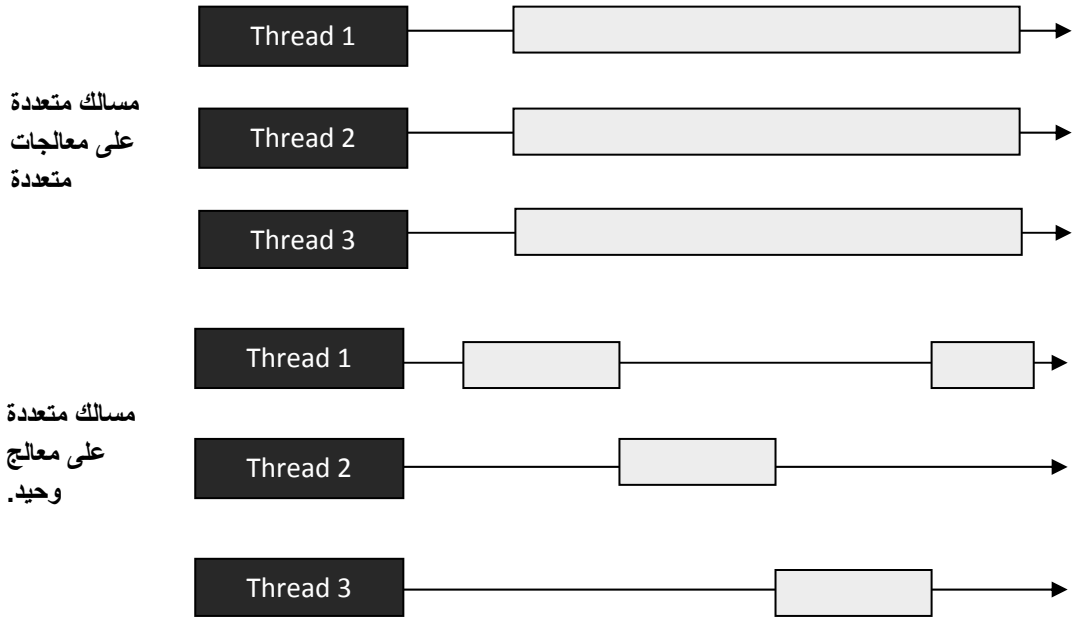
فهو الوحدة الأساسية لتخصيص العمل في نظام التشغيل وكذلك هي تمثل وحدة الكود (مجموعة من التعليمات) التي يمكن أن تنفذ بواسطة المعالج CPU[7,8,13].

إذا المسار هو عملية خفيفة الوزن وهو الأساس لكل العمليات ضمن وحدة المعالجة فكل عملية تحوي مسار واحد على الأقل وهذه المسارات ضمن العملية تتشارك الكود والمراكم والبيانات الموجودة في العملية.

والمسارات التي تكون ضمن عملية ما تتشارك قطاع الكود code section وقطاع البيانات data section وكذلك الموارد مثل الملفات المفتوحة الإشارات والذاكرة المخصصة من الكومة عن طريق malloc.

كل مسار له العناصر المميزة والفريدة الآتية: thread id والمسجلات register وفضاء المكدس stack space [7,8].

قد نستخدم المسارات المتعددة في بنية حاسوبية تحوي على عدة معالجات multiprocessor أو نستخدمها في بنية تحوي على معالج وحيدوهذا مايوضحهالشكل-1-الفرق بين استخدام المسارات المتعددة في كلا الحالتين. وبالتالي لابد لنا من التفريق بين التوازي والتزامن فالتوازي هو مجموعة من الأنشطة التي تحدث في نفس الوقت وكمثال عنها أربع مهام يتم تنفيذها على أبع معالجات في نفس الوقت أما التزامن فهو القدرة على التشغيل في نفس الوقت وكمثال عنه أربع مهام يتم تقسيم الوقت بينها وتنفذ على معالج واحد وفي هذه الحالة نطلق على هذا النوع بالتوازي الوهمي فالتزامن يحوي حالة التوازي الحقيقي وحالة التوازي الوهمي(كلتا الحالتين)[9..13].



الشكل-1-الفرق بين استخدام مسارات المتعددة والمسار الوحيد[7]

النتائج والمناقشة:

1-تصميم خوارزمية نقل الملفات بالاعتماد على المسارات المتعددة

1-1تحديد الحجم الأنسب لعملية النقل:

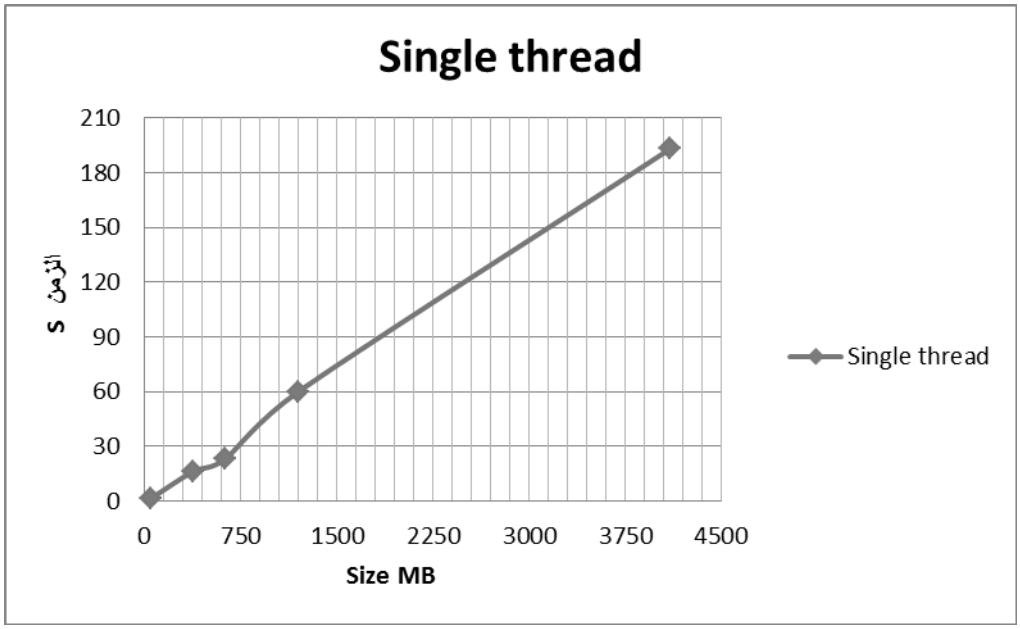
نتيجة لتوزيع الملفات المراد نقلها على مجموعة مسارات معالجة لابد من تحديد الحجم الأعظمي للملفات التي يمكن أن تفرز كمسار معالجة واحد، وهذا الحجم لا يتم حسابه من معادلات بل يتم اختياره لعدة اعتبارات، منها سرعة المعالج أي معدل النقل للمعالج أو بمعنى آخر سرعة النقل بالإضافة إلى الزمن المناسب لعملية النقل.

طبعاً سرعة النقل تحدد من مواصفات المعالج، أما زمن النقل فيحدد من قبل المستخدم أو من اختيار مجموعة أحجامٍ متنوعةٍ ومراقبةٍ زمن النقل لكل منها، ومن خلال هذه الأزمنة نحدد الزمن الأنسب لعملية النقل ومن ثم تحديد الحجم الأعظمي للملفات.يبين الجدول (1)نتائج مراقبة أزمنة نقل مجموعة ملفات بأحجام مختلفة قمنا باختبارها على معالج Core I3 ونظام تشغيل Win7 باستخدام مسار معالجة واحد أو بتنفيذ عمليات النقل المعتادة التي يدعمها نظام التشغيل (نسخ – لصق) أو (قص – لصق).

جدول (1) أزمنة نقل مجموعة من الملفات بأحجام مختلفة

حجم الملفات	زمن النقل
43 MB	1.2 s
374 MB	16s
519 MB	23 s
1.5 GB	60 s
4.1 GB	193 s

من خلال النتائج السابقة نستطيع رسم التابع بين حجم الملفات والزمن المستهلك لعملية النقل كما هو موضح في الشكل (2).



الشكل (2) منحني العلاقة بين حجم الملفات وزمن النقل

يوضح المنحني بأن العلاقة شبه خطية ومن النتائج يمكن اختيار الحجم الأعظمي للملفات بآلية تناسب من حيث الزمن رغبة المستخدم وكلما كان الزمن أقصر كان عدد المسارات المستخدمة أكبر لأن الحجم الأعظمي لملفات المسار الواحد سيكون أصغر لذلك اخترنا الحجم 750 MB كحجم أعظمي لقائمة الملفات باعتباره الزمن المناسب من حيث عدد المسارات والزمن المستهلك ويقابل زمن حوالي 30 sec وهو زمن مقبول كزمن نقل أعظمي.

2-1 تصميم الغرض "قائمة الملفات":

يتوجب علينا الاستفادة من إمكانيات البيئات البرمجية الحديثة لتحقيق بنية برمجية جيدة وذلك من خلال الاعتماد على البرمجية غرضية التوجه في بناء الأغراض اللازمة.

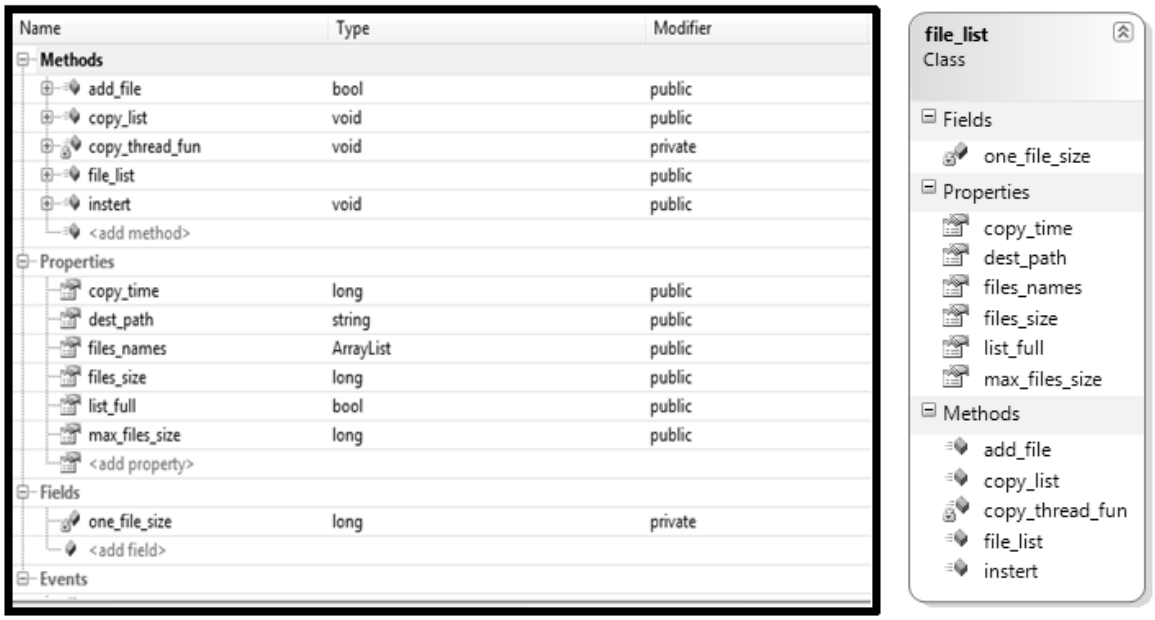
ومن خلال هذا البحث نحتاج لبناء غرض باسم قائمة الملفات File List ليحتوي بداخله على المسارات الخاصة بمجموعة من الملفات تشكل بمجموعها حجما مناسباً لعملية النقل من خلال مسار معالجة واحدة.

وباعتبار أن كل قائمة سيتم تخصيص مسار معالجة خاص بها لذلك سنضمن الغرض بالتوابع اللازمة لإنشاء المسار ضمن الغرض نفسه ليتم استدعاؤه فور جاهزية توزيع الملفات لعملية النقل.

لا بد في البداية من تحديد الخصائص properties والطرق methods اللازمة ضمن هذا الغرض وهذا ما يتم تحديد بعد دراسة بسيطة لمكونات هذا الغرض، فهو يجب أن يحتوي على معلومات عن الحجم المخزن ضمنه حالياً بالإضافة إلى مؤشر امتلاء والحجم الأعظمي الممكن تخزينه، بالإضافة إلى المسار الهدف لعملية النقل والزمن المستهلك لهذه العملية.

أما في التوابع اللازمة فنحتاج إلى تابع إدخال ضمن القائمة يجري اختباره للتأكد من إمكانية إدخال الملف ويعيد مؤشر منطقي عن نجاح العملية أو فشلها، بالإضافة إلى تابع إدخال قسري بغض النظر عن الحجم في حال تواجد ملف ذي حجم أكبر من الحجم الأعظمي المقترح، ولابد من وجود تابع خاص لعملية النسخ يتولى عملية إنشاء مسار

المعالجة لتنفيذ عملية النقل بالإضافة إلى إنهاء عمل المسار مع مراقبة الزمن المستهلك لعملية النقل. يوضح الشكل –2–مخطط UML الخاص بغرض قائمة الملفات File List.



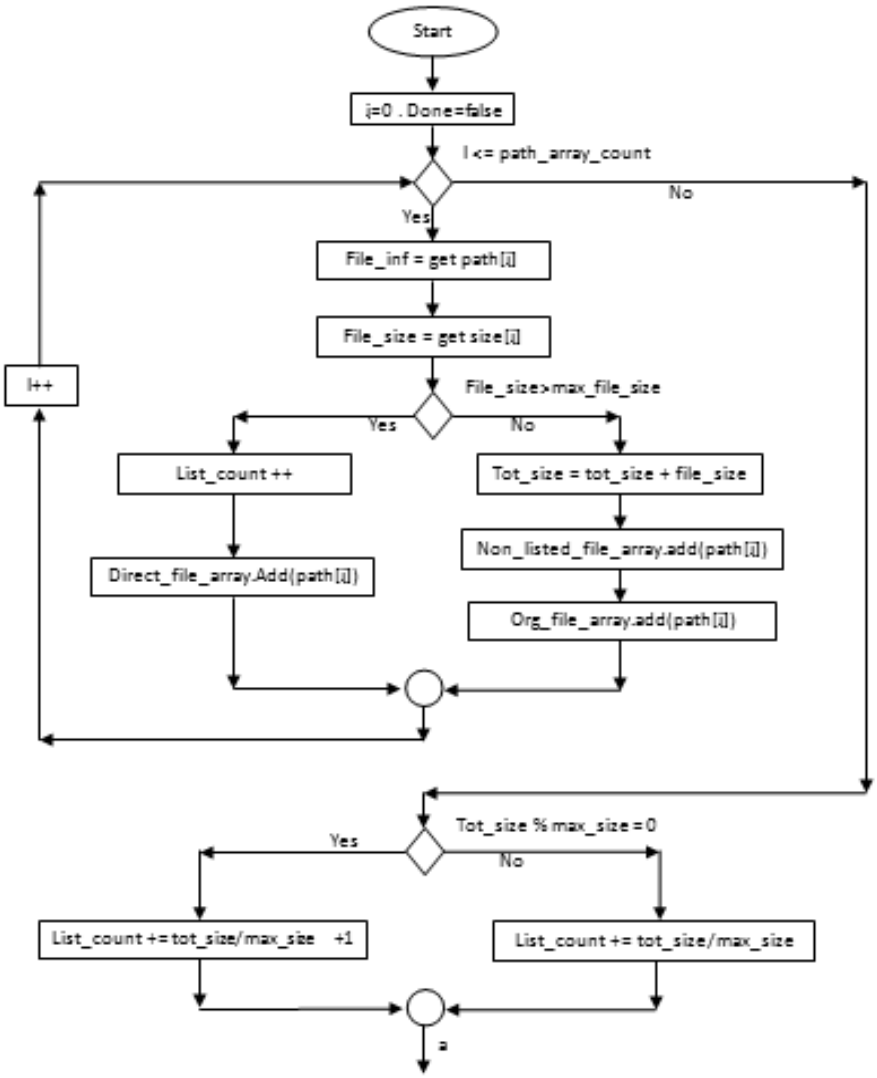
الشكل –2– مخطط UML للغرض File_List مع مكوناته

3–1 تصميم خوارزمية توزيع الملفات:

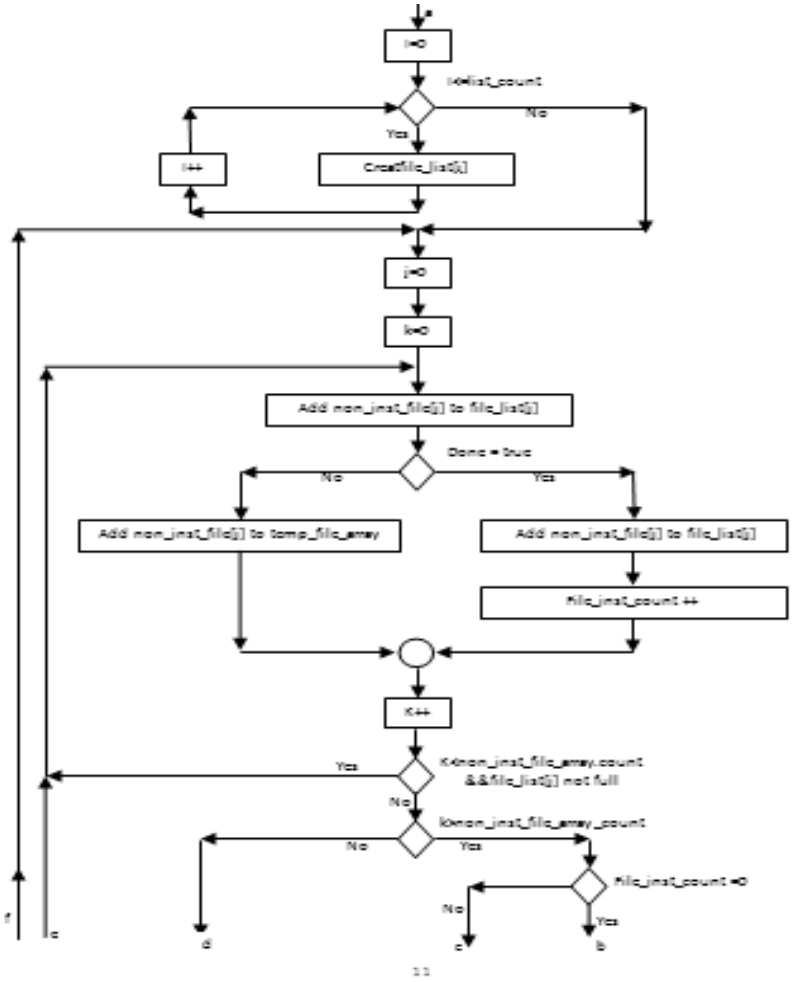
تتولى هذه الخوارزمية عملية توزيع الملفات على مصفوفة قوائم ملفات لذلك لابد من تحديد عدد هذه القوائم بالإضافة إلى فرز الملفات ذات الأحجام الأكبر من الحجم الأعظمي للقائمة ومن ثم البدء بعملية التوزيع من خلال عمليات مرور متكررة على هذه القوائم بهدف تحديد القوائم التي لم تمتلئ بعد في محاولة لتقريب الأحجام قدر الإمكان من الحجم الأعظمي للقائمة؛ وذلك بهدف توزيع هذه الملفات على أقل عدد ممكن من القوائم.

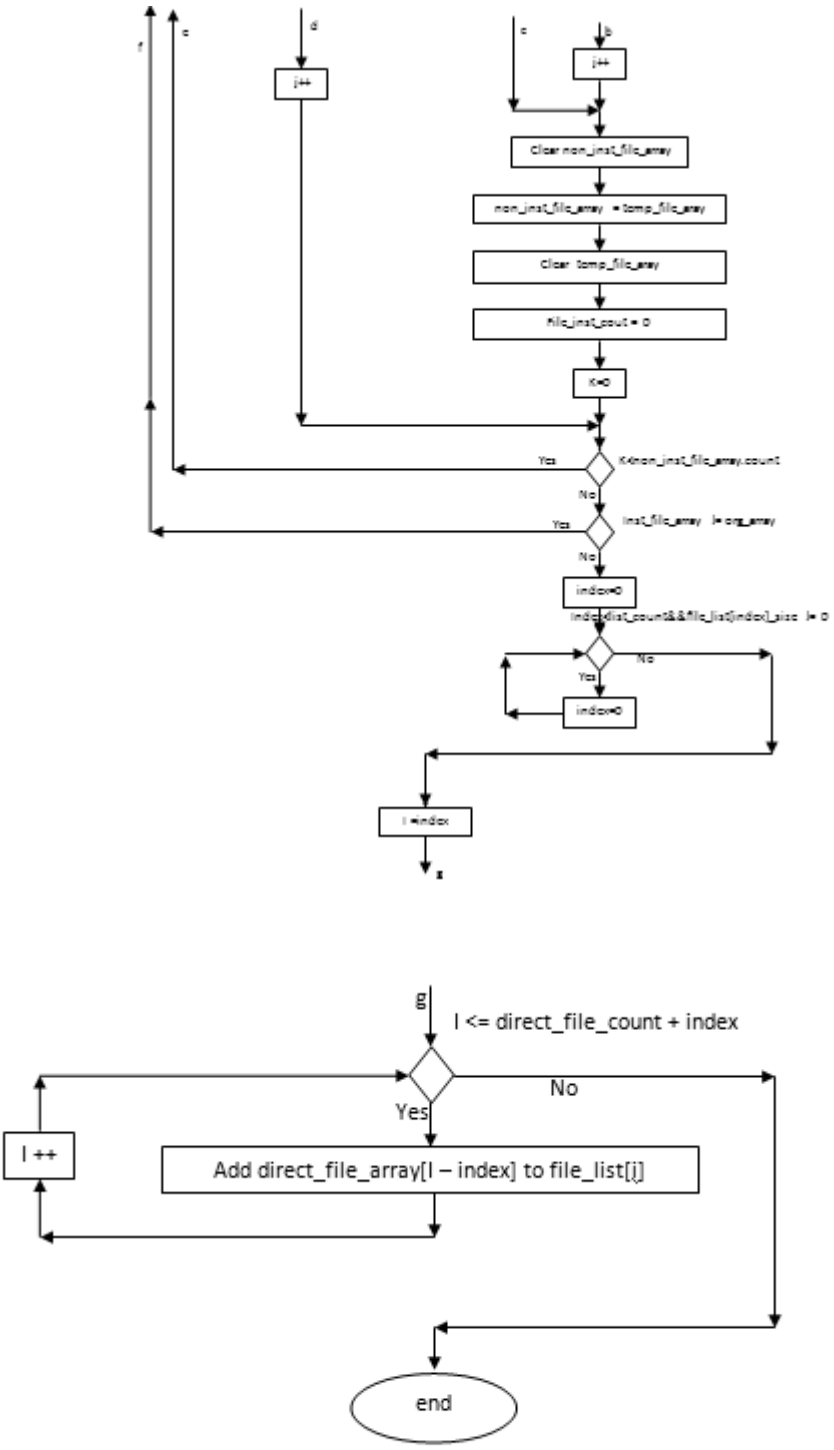
عملية بناء هذه الخوارزمية تتطلب الاعتماد على مجموعة لوائح مساعدة وذلك ليتم تمييز الملفات التي تم توزيعها لتجنب تكرار توزيع الملف على أكثر من قائمة، بالإضافة إلى تحديد الملفات ذات الأحجام الكبيرة لتتم إضافتها بشكل آلي إلى كل ملف ضمن قائمة دون مرورها ضمن الحلقة الرئيسية للتوزيع مع الانتباه لضرورة حذف أحجام هذه الملفات من الحجم الخاضع للتوزيع وذلك لمنع بناء قوائم إضافيةٍ واستهلاكٍ ذاكريٍّ غير مدروس.

يبين الشكل –3– خوارزمية توزيع هذه الملفات مع الأخذ بعين الاعتبار أن زمن عملية التوزيع يعدُّ جزء من زمن النقل الكلي علينا عدم إهماله أثناء عملية المقارنة مع عملية النقل ذات المسار الواحد.



ملاحظة: التابعان get_path و get_size هما من التوابع الجاهزة ضمن لغة البرمجة تعطي معلومات عن الحجم والمسار للملف.


i=0
i<=list_count
No
Yes
i++
Creatfile_list[i]
j=0
k=0
Add non_inst_file[j] to file_list[i]
Done = true
No
Yes
Add non_inst_file[j] to temp_file_array
Add non_inst_file[j] to file_list[i]
File_inst_count ++
K++
K<non_inst_file_array.count
&&file_list[i] not full
Yes
No
k>non_inst_file_array_count
No
Yes
File_inst_count =0
No
Yes
f
e
d
c
b

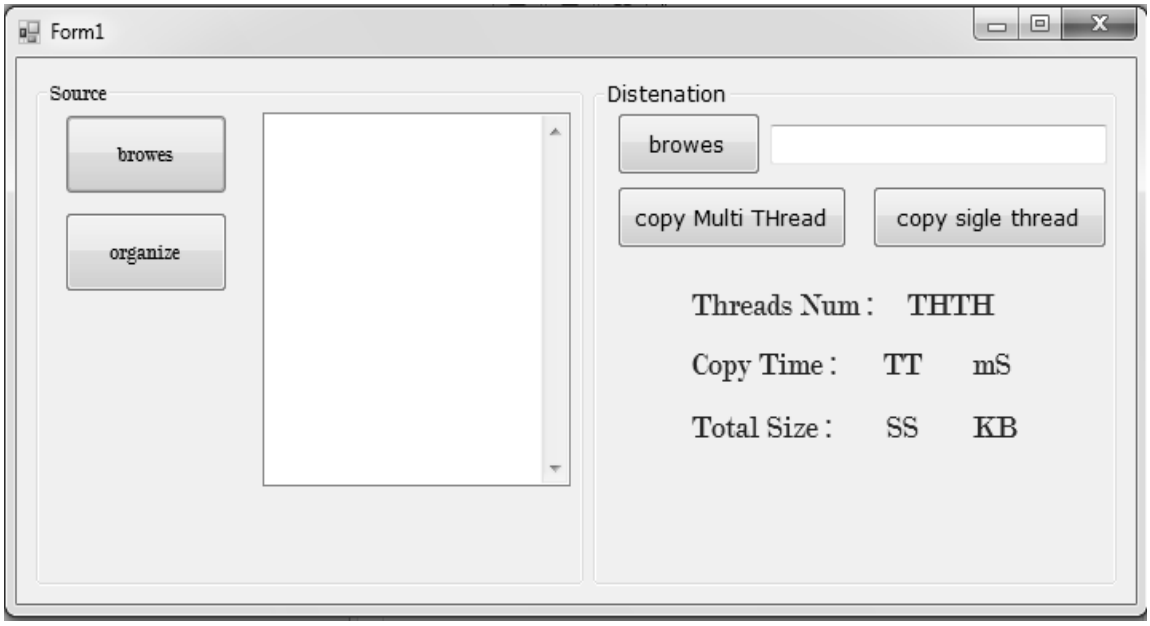


الشكل –3– خوارزمية توزيع الملفات.

4-1 تنفيذ هذه الخوارزمية من خلال بيئة البرمجة net.:

إن بيئة net. بيئة برمجية جيدة في تصميم البرامج ذات الاعتماد الكبير على البرمجة غرضية التوجه بالإضافة إلى وجود بيئة مرئية متميزة نسبياً بين بيئات البرمجة الأخرى[3,4].

بعد أن قمنا بتصميم الخوارزمية لابد من تنفيذها واختبارها، لذلك جهزنا برنامجاً من أجل عمليتي نقل إحداهما تتم دون استخدام المسارات المتعددة، والأخرى تتم من خلال استخدام المسارات المتعددة مع حساب الزمن اللازم لكل من العمليتين الشكل -4- يبين هذه الواجهة.



الشكل -4- واجهة البرنامج الخاصة بالتنفيذ والاختبار

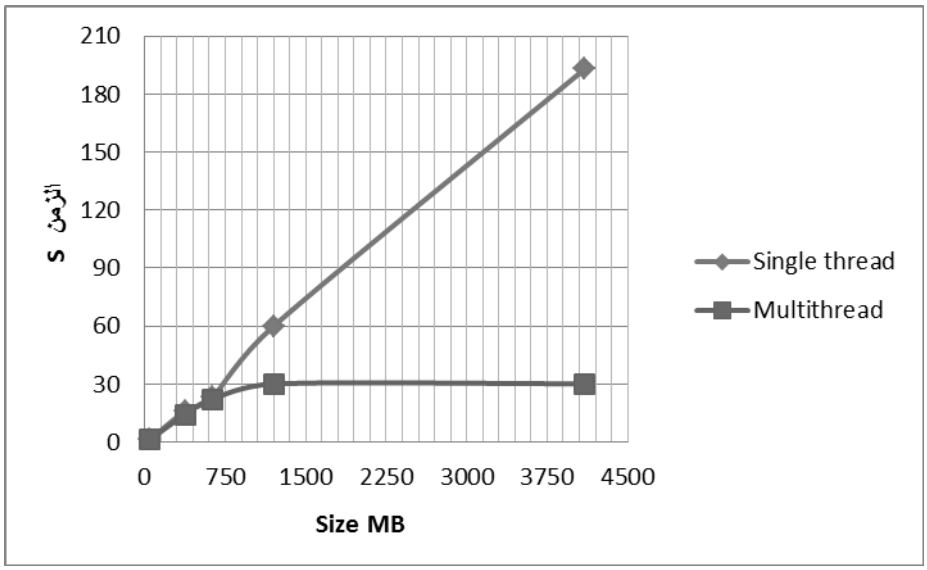
أجرينا على هذه الواجهة مجموعة اختبارات لعمليات نقل ملفات بالآليتين كلتيهما وتم تسجيل الأزمنة المستهلكة لكل منها، ونظمنا هذه النتائج في الجدول 2.

جدول (2) نتائج الاختبار لعمليات النقل من خلال مسار واحد أو عدة مسارات

حجم الملفات	الزمن المستهلك بمسار واحد	الزمن المستهلك بالمسارات المتعددة
43 MB	1.2 s	1.15s
374 MB	16s	14.3S
628 MB	23 s	22.2s
1.2 GB	60 s	30.1 s
4.1 GB	193 s	30.2 s

اخترنا الحجم الأول للنقل أقل من الحجم الأعظمي للمسار الواحد بهدف المقارنة بين الآليتين في حال كانتا بنفس عدد المسارات أي وجود مسار واحد وذلك بهدف تحديد الزمن المستهلك من قبل خوارزمية التوزيع وتحديد مدى

الزيادة في الزمن الناتجة عن هذه الخوارزمية، أما باقي الحالات ففيها الأحجام أكبر من الحجم الأعظمي، وفيها تتزايد الأحجام من أجل زيادة عدد المسارات وذلك بهدف تحديد معامل الإنقاص، ومن أجل ذلك تم توضيح النتائج في المنحنيين 1-2 في الشكل -5-.



الشكل -5- نتائج الاختبار لعمليات نقل الملفات مع وبدون استخدام الخوارزمية المصممة

ملاحظة: للحصول على نتائج صحيحة يجب عدم تكرار الملفات نفسها لأن ذلك يؤثر على النتائج، وفي حال استخدام نفس الملفات يفضل إعادة إقلاع الحاسب بهدف تفريغ الحافظة من الملفات.

وكما هو ملاحظ في الشكل -5- فإن عملية نقل الملفات من خلال مسار واحد تتطلب زمناً يتزايد مع تزايد حجم الملفات بينما أخذت عملية نقل الملفات زمناً أعظمياً ثابتاً في حال استخدام المسارات المتعددة وهو الزمن الذي بنينا وفقه الخوارزمية أي 30Sec.

الاستنتاجات والتوصيات:

من هذا البحث نستطيع استخلاص النتائج الآتية:

- وجود مسارات معالجة متعددة تساعد في تسريع عملية نقل الملفات.
- في الملفات ذات الحجم الأكبر من الحجم الأعظمي لقائمة الملفات لا يمكن اختصار زمن النقل لأن الملف المنقول هو ملف وحيد ولا يمكن تجزئة نقله من خلال هذا التطبيق.
- وجود فروق بسيطة في زمن النقل للملفات ذات الأحجام الأقل من الحجم الأعظمي للقائمة الذي ينتج عن رفع أهمية مسار المعالجة الخاص بالنقل ومن ثم تخصيص المعالج أهمية أعلى لعملية النقل.
- لا يؤثر الزمن المستهلك لعملية توزيع الملفات على القوائم لا يؤثر على زمن النقل الكلي مقارنة مع النقل بمسار معالجة واحد.
- ثبات زمن النقل الأعظمي للملفات بقيمة موافقة لزمن نقل المسار الواحد.
- العلاقة عكسية بين زمن النقل الأعظمي وعدد مسارات المعالجة.

- من خلال الخوارزمية المصممة يمكن تأمين عمليات النقل بشكل متوازن من حيث الحمل على المعالج ، لأنها تعتمد حجماً أعظمياً لمجموعة الملفات المنقولة ضمن مسار المعالجة الواحد.

- تعد هذه الخوارزمية، ومن خلال النتائج المبينة في الجدول –2– ،قابلةً لاعتمادها ضمن الوحدات البرمجية الأساسية الخاصة بنظم التشغيل.

المراجع:

[1] Damien Le Moal, Donald Molaro, Jorge Campello ,*" A Real-Time File System for Constrained Quality of Service Applications"*,Information and Media Technologies , 2010,Vol. 5 (2010) No. 2 pp. 443-458.

[2] Huan Wang; Wen Song, *"Modeling and Analysis of Files transmitting based on Extended Colored Petri nets"*, Journal of Convergence Information Technology . Feb2012, Vol. 7 Issue 2, p174-183. 10p.

[3]Cameron Hughes, Tracey Hughes "*Object-oriented multithreading using C++"*Wiley Computer Pub., 1997 - Computers - 495 pages.

[4] Osni Marques, Michel Dayde, *"High Performance Computing for Computational Science"*, Springer,470,2010.

[5] Richard H.Carver, Kuo-Chung TAI," *Modern Multithreading Implementing, Testing And Debugging Multithreaded Java And C++/PThread/Win32 Program"*, Wiley, 465 , 2006.

[6]Edward L.Lamie, "Real-Time Embedded Multithreading Using ThreadxAnd MIPS", Elsevier, 488, 2009.

[7]Robert A.Lannucci, " *MultiThread Computer Architecture"*, Kluwer Acadmic, 400, 1994.

[8]Bil Lewis, Daniel J.Bery," *MultiThreaded Programming With Java Technology*", Sun, 495, 1997.

[9] Junfeng Yang, Heming Cui, "*Making Parallel Programs Reliable with Stable Multithreading*", Columbia University,2013.

[10] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick. "*A view of the parallel computing landscape.*",Commun. ACM, 52(10):56–67, Oct.2009.

[11] A. Aviram, S.-C.Weng, S. Hu, and B. Ford."*Efficient system-enforced deterministic parallelism*." Commun. ACM, 55(5):111–119, May 2012.

[12] E. A. Lee. "The problem with threads. Computer,ACM, 39(5):33–42, May 2006..

[13]جبر حنا ، انقاص أزمنة الاستعلام ضمن قواعد البيانات اعتماداً على المسارات المتعددة للمعالجة ، مجلة جامعة تشرين 2012.