جدولة موفرة للطاقة ومتسامحة مع الأعطال للتطبيقات المتوازية في أنظمة الوقت الحقيقي الموزعة

طالب الدكتوراه: بهاء الحمدان كلية الهندسة المعلوماتية – جامعة البعث

اشراف الدكتور: ماهر عباس + د. محسن عبود

الملخص

تعتبر قضية تخفيض الطاقة المستهلكة أحدى أهم قيود تصميم الأنظمة المضمنة، ومن أجل ذلك قدم الباحثون تقنية " تدرج الجهد والتردد الديناميكي (DVFS)" والتي تعتبر من أشهر تقنيات خفض الطاقة المستهلكة وأكثرها شيوعاً . حيث تعمل هذه التقنية على خفض تردد المعالج المستخدم في تنفيذ المهمة، وبالتالي تخفيض الطاقة المستهلكة. لكن عملية خفض التردد هذه أدت إلى أضعاف موثوقية النظام من خلال زيادة الأخطاء العابرة التي يتعرض لها المعالج. وهذا شكل تحدياً جديداً لأن الموثوقية قضية جوهرية وأساسية في أنظمة الزمن الحقيقي الحرجة للسلامة، وتنفيذ تطبيق بشكل غير موثوق قد يؤدي إلى عواقب وخيمة. تهدف هذه الدراسة لتنفيذ جدولة موفرة للطاقة ومتسامحة مع الأخطاء لتطبيق متوازي على أنظمة الزمن الحقيقي الموزعة غير المتجانسة، حيث يتم وصف التطبيق المتوازي بواسطة رسم بياني لا يحتوي حلقات موجه (DAG). لتحقيق ذلك تقترح هذه الدراسة خوارزمية جدولة مهام نظام زمن حقيقي موفرة للطاقة مع تحقيق متطلبات الموثوقية (EESARR). للتحقق من كفاءة وقدرة الخوارزمية المقترحة، تمت مقارنة الخوارزمية المقترحة مع خوارزمية (ESRG)، التي تسعى لنفس الهدف لكن بأسلوب مختلف. تؤكد النتائج التجريبية أن استهلاك الطاقة انخفض بواسطة خوارزمية EESARR المقترحة بشكل أكبر من خوارزمية ESRG المدروسة.

الكلمات المفتاحية: رسم بياني لا يحتوي حلقات موجه (DAG)، تدرج الجهد والتردد الديناميكي (DVFS)، كفاءة الطاقة، الجدولة المتسامحة مع الأخطاء، الأنظمة الموزعة غير المتجانسة.

Energy-efficient Fault-tolerant Scheduling of Parallel Applications on Distributed Real-time Systems

Abstract

The issue of reducing energy consumption is one of the most important limitations of the design of embedded systems, and for this researchers introduced the technique of "dynamic voltage and frequency Scaling (DVFS)", which is considered one of the most popular technologies for reducing energy consumption. As this technology works to reduce the frequency of the processor used to execute the task, and thus reduce the energy consumed. However, this downscaling reduced the reliability of the system by increasing the processor's transient errors. This poses a new challenge because reliability is a fundamental issue in real-time safety- critical systems, and implementing unreliable application can lead to dire consequences. This paper aims to implement energy-efficient fault-tolerant scheduling for a reliable parallel application on heterogeneous distributed real-time systems, where the parallel application is described by a directed acyclic graph (DAG). To achieve this, this paper proposes an energy-efficient real-time system tasks scheduling algorithm while meeting reliability requirements (EESARR). To verify the efficiency and capability of the proposed algorithm, The proposed algorithm has been compared with the ESRG algorithm, which seeks the same goal in a different manner. The experimental results confirm that the energy consumption was reduced by the proposed EESARR algorithm more than the studied ESRG algorithm.

Keywords :directed acyclic graph (DAG), dynamic voltage and frequency scaling (DVFS), energy-efficient, fault-tolerant scheduling, heterogeneous distributed systems.

1- مقدمة

مع تطور التطبيقات الحديثة وازياد درجة تعقيدها، كان لا بد من تطور الأنظمة الحاسوبية واتخاذها مساراً مختلفاً بهدف تلبية المتطلبات الحسابية لهذه الأنظمة [1]. حيث ظهرت أنظمة المعالجات المتعددة والتي قدمت تطوراً هائلاً من حيث تحسين الأداء والقوة الحسابية للأنظمة الحاسوبية، لكن هذا التحسين أدى إلى ظهور قضايا ومشاكل جديدة، فبغض النظر عن زيادة الكلفة المادية، ظهرت قضية هامة وهي قضية استهلاك الطاقة، والتي أصبحت تعتبر أحد قيود التصميم الأساسية بالنسبة لأنظمة الحاسوب عامةً والأنظمة المضمنة خاصةً [3]. قضية الطاقة المستهلكة جذبت اهتمام الباحثين في العقد الأخير ليس بسبب الكلفة المادية المترتبة عن زيادة استهلاكها فقط، بل أيضاً بسبب تأثيرها على أداء النظام نفسه. من أجل ذلك قدم الباحثين العديد من التقنيات في سبيل تخفيض استهلاك الطاقة وخصوصاً في الأنظمة المضمنة التي تعتمد على مورد طاقة محدود، لعل أشهر هذه التقنيات هي تقنية DVFS (Dynamic Voltage and Frequency Scaling) والمستخدمة من قبل مصنعي المعالجات الدقيقة حالياً مثل Intel, ARM, and AMD. وعلى الرغم من نجاح هذه التقنية على صعيد توفير الطاقة الديناميكية المستهلكة، إلا أن هذه العملية قد تؤدي إلى إضعاف موثوقية الأنظمة بسبب الارتفاع الحاد في نسبة الفشل العابر"transient failures " للمعالجات التي قد تسببه [2,3-5].

بالنسبة لأنظمة الزمن الحقيقي من النوع الحساس لقضية للسلامة تعتبر الموثوقية من القضايا الأساسية والحرجة التي لا يمكن تجاهلها، فعلى الرغم من أن التسامح مع الأعطال القائم على النسخ المتماثل آلية مهمة لتعزيز الموثوقية [1,2]، لا يمكن أن يكون أي تطبيق موثوقًا به بنسبة 100٪ في الممارسة العملية. لذلك إذا كان التطبيق يفي بهدف الموثوقية المعتمد، فإنه يعتبر موثوقًا به وفقًا لمعايير السلامة الوظيفية، مثل ISO 26262 لأنظمة السيارات [26] ، DO-178B لأنظمة إلكترونيات الطيران، و IEC 61508 لجميع أنواع أنظمة البرامج الصناعية [2].

لكن استخدام التسامح مع الأعطال بهدف تلبية متطلبات موثوقية التطبيق سيسبب زيادة في كمية الطاقة المستهلكة من قبل المعالجات لتنفيذ مهام التطبيق، لذلك يجب الانتباه جيداً إلى علاقة الترابط بين الموثوقية والطاقة المستهلكة [4]. من ناحية، يمكن تأمين متطلبات و هدف الموثوقية من خلال التسامح مع الأعطال لضمان أن التحكم بالمخاطر عند مستوى مقبول لتطبيق السلامة الحرجة. من ناحية أخرى، لا يأتي التسامح مع الأعطال بكلفة مجانية ويتضمن عمومًا نفقات إضافية من موارد النظام أبرزها الطاقة [4].

بالإضافة إلى ما سبق يجب الانتباه بحذر أيضاً إلى أن استخدام تقنية النسخ المتماثل النشط من أجل ضمان الموثوقية، سيؤدي إلى زيادة عدد نسخ المهام المستخدمة وبالتالي زيادة الوقت اللازم لتنفيذ هذه النسخ، كما أن استخدام تقنية DVFS من اجل توفير الطاقة المستهلكة ستؤدي إلى خفض التردد المستخدم وبالتالي زيادة زمن التنفيذ، لكن في انظمة الزمن الحقيقي الصارمة من النوع الحساس لقضية السلامة تجاوز القيد الزمني المفروض على التطبيق قد يؤدي إلى حدوث كوارث[2]، وبناءً على ما سبق يمكننا الاستنتاج بوضوح أن عملية تحسين أي من العوامل الثلاثة في جدولة المهام (الوقت، الموثوقية والطاقة) يؤدي إلى إضعاف العاملين الباقيين، وعليه أن أي عملية تحسين لأي عامل من العوامل الثلاثة يجب أن تتم مع مراعاة العاملين الباقيين، وهنا تكمن صعوبة المشكلة.

2- الغرض من البحث

الهدف الرئيسي من هذا البحث هو العمل على بناء خوارزمية جدولة مهام نظام زمن حقيقي صارم ، تعمل هذه الخوارزمية على جدولة مهام تطبيق يتألف من عدد من المهام المرتبطة ببعضها بعلاقة أسبقية، ويتم تمثيل هذا التطبيق من خلال مخطط DAG(Directed Acyclic Graph)، تعمل خوارزمية الجدولة هذه على ضمن بيئة معالجات غير متجانسة بهدف إيجاد أفضل عملية جدولة ممكنة لمهام التطبيق، مع تلبية متطلبات موثوقية هذا التطبيق، وتخفيض الطاقة الديناميكية المستهلكة من قبل مهام هذا التطبيق، مع ضمان عدم تجاوز القيد الزمني المفروض على التطبيق.

3- مفهوم الحوسبة المتسامحة مع الأعطال

الحوسبة المتسامحة مع الأعطال تعني الحوسبة بشكل صحيح على الرغم من وجود أخطاء في النظام. بالنسبة لأنظمة المعالجات المتعددة فإن تقنية تكرار المهام تعد من أهم تقنيات التسامح مع الأعطال الشائعة:

- تكرار المهام (Task Replication):

تعتبر هذه التقنية من أكثر التقنيات البرمجية للتسامح مع الأعطال. فكرة هذه التقنية هي في حال فشل المعالج بتنفيذ مهمة ما فإن ذلك يجب ألا يؤدي إلى فشل النظام ككل، وذلك من خلال توافر نسخة احتياطية أو أكثر يتم تنفيذها في حال فشل المهمة الرئيسية [10] ، يوجد نوعين رئيسين من هذه التقنية وهما:

- ✓ النسخ المتماثل السلبي (Passive Replication Schema):

في هذا النوع يقوم المجدول بإنشاء نسخة احتياطية من كل مهمة من مهام التطبيق، لكن لا يقوم بإسنادها إلى معالج من المعالجات المتاحة، إلا في حال حدوث فشل في تنفيذ المهمة الأساسية المسندة [10].

- ✓ النسخ المتماثل النشط (Active Replication Schema):

يقوم المجدول في هذا النوع بإنشاء نسخة واحدة أو أكثر من كل مهمة من مهام التطبيق، ويتم تنفيذ هذه النسخ مع المهمة الرئيسية بشكل متزامن وبنفس اللحظة الزمنية، فإذا حدث فشل في تنفيذ اي المهمة الرئيسية أو نسخها فإنه سيتم تنفيذ المهمة مرة واحدة على الأقل [4].

4- مفهوم الطاقة وأهم تقنيات تخفيض استهلاك الطاقة

يشير مفهوم الطاقة إلى الكمية القياسية التي تصف قدرة العمل الذي يمكن أن يؤديه النظام [6]. وتعتبر عملية استهلاك الطاقة قضية أساسية في تصميم الأنظمة المدمجة التي تعاني من محدودية الموارد مثل الطاقة المحدودة أو تلك التي تعتمد على البطاريات كمصدر للطاقة. ولعل أشهر تقنيات تخفيض استهلاك الطاقة على مستوى المعالجات تقنية تدريج التردد والجهد الديناميكي Dynamic Voltage Frequency Scaling (DVFS). حيث تعمل هذه التقنية على تخفيض الطاقة الديناميكية المستهلكة من قبل المعالجات من خلال تخفيض التردد (أي الجهد المستخدم) ذلك عن طريق ضبط تردد المعالج حسب الاحتياجات الفعلية، من أجل الحفاظ على الطاقة. تساعد تقنية DVFS في الحفاظ على البطارية على الأجهزة المحمولة [6] وتقليل تكلفة التبريد نتيجة تخفيض درجة الحرارة إضافة لخفض الطاقة الديناميكية المستهلكة. لكن مع ملاحظة أن استخدام تقنية DVFS، تؤدي غلى زيادة زمن تنفيذ المهام بسبب عملية خفض التردد المستخدم لتنفيذ المهام كما أنها تؤدي إلى أضعاف موثوقية النظام، وكلا القضيتين تعتبران قضايا محورية وجوهرية في تصميم أنظمة الزمن الحقيقي الحرجة للسلامة، لذلك يجب توخي الحذر عند استخدام هذه التقنية.

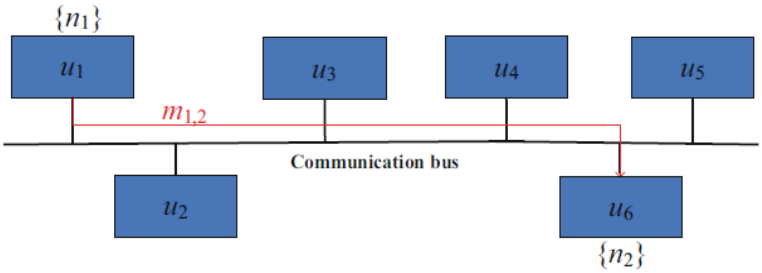
ملاحظة: بما أن العلاقة بين الجهد والتردد علاقة خطية، حيث أن زيادة التردد تؤدي إلى زيادة الجهد، والعكس صحيح، سنقوم باستخدام مصطلح التردد للإشارة إلى كلا المصطلحين التردد والجهد أي عندما نقول خفض التردد فهذا يعني خفض كل من التردد والجهد المستخدم والعكس صحيح.

5- بنية النظام System Architecture:

قمنا باختيار بنية معمارية موزعة شائعة جداً [2] لتمثيل النظام، حيث يتكون النظام من مجموعة من المعالجات يتم تركيبها على نفس الناقل، أي أن الناقل يكون مشترك بين جميع المعالجات. كما هو موضح في الشكل 1. يحتوي كل معالج على وحدة معالجة مركزية وذاكرة وصول عشوائي (RAM) وذاكرة غير متطايرة، وبطاقة واجهة الشبكة. يتم تنفيذ

المهمة بالكامل على معالج واحد، والذي يقوم بإرسال رسائل نتيجة التنفيذ إلى المهام اللاحقة جميعها، والتي قد تكون موجودة في معالجات مختلفة. على سبيل المثال، يتم تنفيذ المهمة n_1 على المعالج u_1، ثم يرسل رسالة $m_{1,2}$ إلى المهمة اللاحقة n_2 الموجودة في u_6 (الشكل 1).

بفرض U = {u1, u2, . . . , u|U|} تمثل مجموعة من المعالجات غير المتجانسة، حيث |U| تمثل حجم المجموعة U. لأي مجموعة X، نستخدم في هذا البحث التعبير |X| للدلالة على حجمها[3].



الشكل (1) بنية نظام موزع متعدد المعالجات غير متجانس [2].

1-5 نموذج التطبيق المتوازي Parallel Application Model:

يتم تمثيل التطبيق المتوازي من خلال مخطط DAG كما يلي: DAG G = (N,W,M,C). [1,3,4,5]

1. تمثل N مجموعة من العقد في G، وكل عقدة $n_i \in N$ تمثل مهمة من مهام التطبيق. يمثل $\text{pred}(n_i)$ مجموعة المهام السابقة المباشرة للمهمة n_i. أما $\text{succ}(n_i)$ يمثل مجموعة المهام اللاحقة المباشرة للمهمة n_i. المهمة التي ليس لها مهمة سابقة يُشار إليها ب n_{entry} بمعنى مهمة دخل، والمهمة التي ليس لها مهمة لاحقة تدعى مهمة خرج ويُشار إليها ب n_{exit}. إذا كان التطبيق يحتوي على عدة مهام دخل n_{entry} أو عدة مهام خرج n_{exit}، تتم إضافة مهمة دخل أو خرج وهمية

بكلفة تنفيذ صفر وبعلاقات أسبقية قيمتها(وزنها) صفر، بحيث يبدأ التطبيق بمهمة دخل واحدة وينتهي بمهمة خرج واحدة.

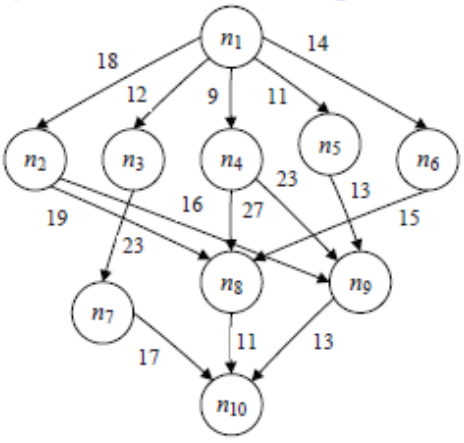
2. W هي مصفوفة |N| × |U|، حيث تشير $w_{i,k}$ إلى وقت التنفيذ بأسوأ الأحوال (Worst Case Execution Time WCET) للمهمة n_i عند عملها على المعالج u_k بأقصى تردد للمعالج. كل مهمة $n_i \in N$ لها قيمة تنفيذ WCET مختلفة على معالجات مختلفة بسبب عدم تجانس المعالجات. WCET للمهمة هي أكبر وقت تنفيذ بين جميع قيم وقت التنفيذ الحقيقي الممكنة عند تنفيذ المهمة على معالج محدد بأقصى تردد. جميع قيم WCETs للمهام معروفة ويتم تحديدها من خلال إجراء طرق التحليل (أي تحليل WCET) خلال مرحلة التحليل.
3. عملية الاتصال بين المهام المسندة إلى معالجات مختلفة يتم من خلال تمرير الرسائل عبر الناقل المشترك. M عبارة عن مجموعة من حواف الاتصال، وكل حافة $m_{i,j} \in$ M تمثل رسالة من n_i إلى n_j . أي أن المهمة n_j هي مهمة لاحقة مباشرة للمهمة n_i، ولا يمكنها أن تبدء التنفيذ قبل وصول خرج المهمة n_i إليها، وفي حال كان للمهمة n_j أكثر من مهمة سابقة مباشرة فيجب عليها أن تنتظر خرج جميع المهام السابقة لها حتى تستطيع بدء التنفيذ.
4. تمثل C مصفوفة كلف نقل الرسائل، أي أن $c_{i,j} \in C$ هي كلفة نقل الرسالة من المهمة n_i إلى المهمة n_j، أي أسوأ وقت استجابة ممكن (Worst Case WCRT Response time) إذا لم يتم اسناد المهمة n_i و المهمة n_j إلى نفس المعالج. WCRT لرسالة هي أكبر وقت استجابة بين جميع قيم وقت الاستجابة الممكنة عندما يتم إرسال الرسالة على عتاد مادي معين(الناقل المشترك). إذا تم تعيين n_i و n_j على نفس المعالج، فإن قيمة وقت WCRT تكون صفر. جميع قيم WCRTs للرسائل تكون معروفة أيضًا ومحددة من خلال طريقة تحليل WCRT خلال مرحلة التحليل.

يوضح الشكل 2 تطبيقًا موازيًا [5−1]. التطبيق المتوازي في المثال يتكون من 10 مهام يتم تنفيذها على 3 معالجات $\{u_1, u_2, u_3\}$. يوضح الجدول 1 مصفوفة WCET للتطبيق في الشكل 2. على سبيل المثال زمن تنفيذ المهمة n_1 على المعالج u_1 هو 14 بأسوأ الأحول ويشار إيها ب $w_{1,1} = 14$. وبما أن المعالجات غير متجانسة بالتالي فإن زمن تنفيذ المهمة n_1 على المعالج u_2 تختلف عن زمن تنفيذ المهمة على المعالج u_1، ويشار إليه ب $w_{1,2} = 16$ ويختلف عن زمن تنفيذ المهمة بأسوأ الأحوال عن المعالج u_3، ويشار إليه ب $w_{1,3} = 9$.

تمثل قيمة الوزن البالغة 18 لحافة الاتصال بين n_1 و n_2 الزمن (الكلفة) اللازم لإرسال رسالة من المهمة n_1 إلى المهمة n_2، ويشار لهذا الزمن ب $c_{1,2} = 18$، ذلك في حال تم اسناد المهمتين n_1 و n_2 إلى معالجين مختلفين، لكن في حال تم اسناد المهمتين لنفس المعالج تكون كلفة الاتصال بينهما $c_{1,2} = 0$.

الجدول (1) قيم WCET للمهام.

Task	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16



الشكل (2) مثال عن تطبيق الموازي بالاعتماد على مخطط DAG [3].

2-5 نموذج الطاقة والاستطاعة Power and Energy Model:

العلاقة بين الجهد والتردد هي علاقة خطية [9]، أي أن تقليل الجهد يؤدي إلى خفض التردد والعكس صحيح، يتم تقليل الجهد والتردد بواسطة تقنية DVFS بهدف توفير الطاقة. على غرار[10,4,2]، في هذا البحث سنستخدم مصطلح تغيير التردد للإشارة إلى التغيير المتزامن في الجهد والتردد. بالنسبة للأنظمة القادرة على تفعيل واستخدام تقنية DVFS، فإن نموذج طاقة النظام المستخدم على نطاق واسع [10,4,2] يستخدم أيضاً في هذا البحث، حيث يتم حساب الطاقة المستهلكة من قبل النظام أثناء استخدام التردد f على النحو التالي [4]:

$$P(f) = P_{\mathrm{s}} + g(P_{\mathrm{ind}} + P_{\mathrm{d}}) = P_{\mathrm{s}} + g(P_{\mathrm{ind}} + C_{\mathrm{ef}} f^{m}) \quad (1)$$

تمثل P_{s} الطاقة الساكنة وهي الطاقة التي تمر عبر الدارات الإلكترونية للنظام (في بحثنا المعالجات) حتى لو لم يكن النظام ينفذ أي عمل أو مهمة، وبالتالي يكفي أن يكون النظام بحالة تشغيل حتى يستهلك هذه الطاقة، ويمكن إزالتها فقط عن طريق إيقاف تشغيل النظام بأكمله.

تمثل P_{ind} الطاقة الديناميكية المستقلة عن التردد، ويقصد بها الطاقة التي تستهلكها الدارات الإلكترونية للنظام أثناء تنفيذ عمل أو مهمة ما بغض النظر عن قيمة التردد المستخدم في النظام.

أما P_{d} فهو يمثل الطاقة المستهلكة الديناميكية المعتمدة على التردد. ويقصد بها الطاقة التي تستهلكها الدارات الإلكترونية للنظام أثناء تنفيذ عمل أو مهمة ما، وتتغير هذه القيمة بتغير قيمة التردد المستخدم في النظام.

يمثل المتحول البولياني g حالة النظام ويشير إلى ما إذا كانت الطاقة الديناميكية يتم استهلاكها حاليًا في النظام. عندما يكون النظام نشط، فإن $g = 1$، أي النظام يقوم بتنفيذ مهمة أو عمل معين. خلاف ذلك فإن $g = 0$. فيما يمثل المتحول C_{ef} السعة أو القدرة التبديلية الفعالة للمعالج أي عند انتقاله من تردد لأخر، ويمثل المتحول m أس القوة الديناميكية، الذي يجب ألا يقل عن 2. كل من المتحولين C_{ef} و m هي ثوابت تعتمد على

المعالج، وبما أن النظام المعتمد في هذا البحث غير متجانس فإن قيمة المتحولين قد تختلف من معالج لأخر. كما قد تختلف الترددات التي يدعمها كل معالج وبالتالي عند تنفيذ نفس المهمة على معالجين مختلفين، فإن قيم الطاقة الديناميكية المستقلة عن التردد P_{ind} والطاقة المستهلكة المعتمدة على التردد P_{d} تكون مختلفة حتى من أجل نفس المهمة.

تكلفة الحمل الزائد (الطاقة المستهلكة) الناتج عن عملية تشغيل/إيقاف تشغيل النظام ليست قليلة وتجعل من استهلاك النظام للطاقة P_{s} أمر لا يمكن إدارته [4]. أيضاً التقنيات المستخدمة في صناعة المعالجات الحديثة جعلت الطاقة الثابتة المستهلكة اقل ما يمكن. على غرار الدراسات السابقة، تركز دراستنا المقدمة على إدارة القوة الديناميكية (أي P_{ind} و P_{d}). بالنظر إلى كون P_{ind} مستقل عن التردد، فإن تخفيض P_{d} لا يعني بالضرورة تقليل استهلاك الطاقة. لذلك، فإن الحد الأدنى للتردد الموفر للطاقة والذي يرمز له ب f_{ee} (energy-efficient frequency) يتم التعبير عنه على النحو التالي [4]:

$$f_{\mathrm{ee}} = \sqrt[m]{\frac{P_{\mathrm{ind}}}{(m-1)C_{\mathrm{ef}}}} \qquad (2)$$

أي بمعنى تخفيض التردد لقيمة أقل من قيمة f_{ee} لن يؤدي إلى تخفيض استهلاك الطاقة، لذلك تعتبر القيمة f_{ee} هي الحد الأدنى المسموح به الذي يمكن أن تعمل به المعالجات. لكن المعالجات تدعم مجموعة محددة من الترددات تتراوح قيمها من f_{min} الذي يمثل أصغر تردد يمكن أن يدعمه المعالج إلى القيمة f_{max} الذي يمثل أعلى تردد يمكن أن يدعمه المعالج، بالتالي فإن أصغر تردد يمكن أن يعمل عليه المعالج ويُشار إليه ب f_{low} يُعطى على النحو التالي[4]:

$$f_{\mathrm{low}} = max(f_{min}, f_{\mathrm{ee}}) \qquad (3)$$

أي أن تخفيض التردد إلى قيمة أقل من قيمة f_{ee} لن يؤدي إلى تخفيض استهلاك الطاقة، كما أن التردد الأصغر الذي يمكن أن يدعمه المعالج هو f_{min}، ولا يمكن تخفيض التردد إلى قيمة أصغر من هذه القيمة لذلك يصبح الحد الأدنى للتردد الذي يمكن أن يعمل عليه المعالج

هو أكبر قيمة بين القيمتين f_{ee} و f_{min}. لذلك فإن أي تردد فعال f_{h} يمكن أن يعمل عليه المعالج يجب أن يكون ضمن المجال التالي $f_{\text{low}} \leqslant f_{\text{h}} \leqslant f_{max}$.

ليكن $E_{\text{d}}(n_i, u_k, f_{k,v})$ يمثل الطاقة الديناميكية المستهلكة نتيجة تنفيذ المهمة n_i على المعالج u_k باستخدام التردد $f_{k,v}$. بالتالي يمكن حساب هذا التعبير كما يلي[5]:

1. مجموعة الترددات المستقلة عن التردد للمعالجات: $\{P_{1,\text{ind}}, P_{2,\text{ind}}, \dots, P_{|U|,\text{ind}}\}$.
2. مجموعة الترددات المعتمدة على التردد للمعالجات: $\{P_{1,\text{d}}, P_{2,\text{d}}, \dots, P_{|U|,\text{d}}\}$.
3. مجموعة أسس الطاقة الديناميكية للمعالجات: $\{m_1, m_2, \dots, m_{|U|}\}$.
4. مجموعة قدرات التبديل الفعالة للمعالجات: $\{C_{1,\text{ef}}, C_{2,\text{ef}}, \dots, C_{|U|,\text{ef}}\}$.
5. مجموعة أخفض تردد يمكن أن يعمل عليه المعالج: $\{f_{1,\text{low}}, f_{2,\text{low}}, \dots, f_{|U|,\text{low}}\}$.
6. مجموعة الترددات الفعالة للمعالجات :

$$\begin{Bmatrix} \{f_{1,\text{low}}, f_{1,a}, f_{1,b}, \dots, f_{1,max}\} \\ \{f_{2,\text{low}}, f_{2,a}, f_{2,b}, \dots, f_{2,max}\} \\ \vdots \\ \{f_{|U|,\text{low}}, f_{|U|,a}, f_{|U|,b}, \dots, f_{|U|,max}\} \end{Bmatrix}$$

بالتالي فإن $E_{\text{d}}(n_i, u_k, f_{k,v})$ تحسب على النحو التالي [5]:

$$E_{\text{d}}(n_i, u_k, f_{k,v}) = \left(P_{k,\text{ind}} + C_{k,\text{ef}} \times (f_{k,v})^{m_k}\right) \times w_{i,k} \times \frac{f_{k,max}}{f_{k,v}} \qquad (4)$$

في هذه الدراسة، الكلفة الناتجة عن تغير التردد تعتبر مقدار ضئيل جداً مقارنتاً بالطاقة الديناميكية المستهلكة ولذلك تهمل [11].

5-3 نموذج الموثوقية والتسامح مع الأعطال :

بالنسبة للأنظمة التي لا تعمل بتقنية DVFS، اقترح Shatz و Wang أولاً أن احتمالية موثوقية المعالج تخضع لتوزيع Poisson [12]. هذه الفكرة مستخدمة على نطاق واسع في

الدراسات السابقة [1,5,10,12]. حيث يُشار إلى موثوقية حدث ما خلال وحدة الزمن t كما يلي [12]:

$$R(t) = e^{-\lambda t} \qquad (5.1)$$

بفرض λ_k يمثل ثابت معدل فشل المعالج u_k خلال وحدة الزمن. بالتالي يتم حساب موثوقية المهمة n_i المنفذة على المعالج u_k باستخدام التعبير التالي [12]:

$$R(n_i, u_k) = e^{-\lambda_k w_{i,k}} \qquad (5.2)$$

في الأنظمة القادرة على تفعيل واستخدام تقنية DVFS، تنتج الترددات المختلفة معدلات فشل مختلفة وفقًا لملخصات الأبحاث ذات الصلة [4,5,10]. بفرض $\lambda_{k,max}$ تمثل معدل فشل المعالج u_k عندما يعمل بأقصى تردد، و $\lambda_{k,v}$ يمثل معدل فشل المعالج u_k باستخدام التردد $f_{k,v}$ ويتم حسابه على النحو التالي[5]:

$$\lambda_{k,v} = \lambda_{k,max} \times 10^{\frac{d(f_{k,max} - f_{k,v})}{f_{k,max} - f_{k,min}}} \qquad (6)$$

حيث d قيمة ثابتة، تمثل حساسية معدلات الفشل لتغير الجهد.

ثم يتم بناء العلاقة بين موثوقية المهمة والتردد الذي سيعمل عليه المعالج الذي سينفذ المهمة حسب المعادلات: (5) و (6) ، أي أن موثوقية المهمة n_i المنفذة على المعالج u_k باستخدام التردد $f_{k,v}$ تحسب على النحو التالي :

$$R(n_i, u_k, f_{k,v}) = e^{-\lambda_{k,max} \times 10^{\frac{d(f_{k,max} - f_{k,v})}{f_{k,max} - f_{k,min}}} \times \frac{w_{i,k} \times f_{k,max}}{f_{k,v}}} \qquad (7)$$

في المعادلة (7) العلاقة بين الموثوقية والتردد تتزايد باستمرار على نفس المعالج .لذلك، تقليل الجهد والتردد بهدف تقليل استهلاك الطاقة الديناميكية ينتج عنه تخفيض موثوقية التطبيق.

أن التقنية المستخدمة في هذا البحث هي تقنية النسخ المتماثل النشط المتغير من أجل التسامح مع الأعطال، وكما ذكرنا سابقاً فإن هذه التقنية تتميز بعدد متغير من النسخ المكررة من أجل كل مهمة، أي أن عدد النسخ من أجل المهمة n_1 يختلف عن عدد النسخ

المستخدمة من أجل المهمة n_2 ، لذلك نعرف المتحول $\mathrm{num}_i(\mathrm{num}_i \leqslant |U|)$ والذي يشير إلى عدد النسخ الخاصة بالمهمة n_i في تقنية النسخ المتماثل النشط. وطالما أنه لا يمكن أسناد نسختين إلى نفس المعالج يكون لدينا مجموعة النسخ الخاصة بالمهمة n_i هي $\{n_i^1, \dots, n_i^\beta, \dots, n_i^{num_i}\}$، حيث n_i^1 هي النسخة الأساسية، وباقي عناصر المجموعة تمثل النسخ الاحتياطية. وبما أن موثوقية المهمة n_i على المعالج u_k تعطى في المعادلة (5). فإن احتمالية فشل المهمة n_i على المعالج u_k تحسب كما يلي [12]:

$$1 - R(n_i, u_k) = 1 - e^{-\lambda_k w_{i,k}} \qquad (8)$$

بالتالي مع استخدام أكثر من نسخة من المهمة n_i تصبح موثوقية المهمة n_i كما يلي [12]:

$$R(n_i) = 1 - \prod_{\beta=1}^{num_i} (1 - R(n_i^\beta, u_{pr(n_i^\beta)}, f_{pr(n_i^\beta),hz(n_i^\beta)})) \qquad (9)$$

حيث يشير المتحول $u_{pr(n_i^\beta)}$ إلى المعالج الذي اسُندت إليه النسخة β من المهمة n_i أي n_i^β، أما المتحول $f_{pr(n_i^\beta),hz(n_i^\beta)}$ فيشير إلى التردد المستخدم من قبل المعالج لتنفيذ النسخة β من المهمة n_i.

بالنظر إلى أن موثوقية التطبيق هي نتاج قيم الموثوقية لجميع مهامه [12,2]، بالتالي فإن قيمة موثوقية التطبيق تحسب كما يلي[12]:

$$R(G) = \prod_{n_i \in N} R(n_i) \qquad (10)$$

الطاقة الديناميكية المستهلكة من قبل المهمة n_i هي مجموع الطاقة الديناميكية المستهلكة من قبل كل نسخة من نسخ المهمة n_i [12]:

$$\mathrm{E_d(n_i)} = \sum_{\mathrm{x=1}}^{\mathrm{num_i}} \mathrm{E_d(n_i^\beta, u_{pr(n_i^\beta)}, f_{pr(n_i^\beta),hz(n_i^\beta)})} \qquad (11)$$

وبالتالي فإن الطاقة الديناميكية المستهلكة من قبل التطبيق هي مجموع الطاقة الديناميكية المستهلكة من قبل جميع مهام التطبيق وتحسب كما يلي [12]:

$$E_{\mathrm{d}}(G) = \sum_{i=1}^{|N|} E_{\mathrm{d}}(n_i) \qquad (12)$$

5-4 توصيف المشكلة:

يمكن وصف المشكلة التي سنقوم بتناولها في هذه الدراسة كالآتي. بفرض أنه لدينا تطبيق متوازي G ، يتألف هذا التطبيق من مجموعة من المهام المترابطة مع بعضها البعض N، ولدينا مجموعة من المعالجات غير المتجانسة U، هذه المعالجات تدعم مستويات تردد مختلفة (تدعم تقنية DVFS). تتضمن المشكلة اسناد نسخ المهمة المتماثلة للمعالج المناسب بالتردد المناسب بحيث يتم تقليل إجمالي الطاقة المستهلكة قدر الإمكان، والتأكد من أن الموثوقية التي تم الحصول عليها للتطبيق $R(G)$ أكبر من أو تساوي هدف الموثوقية $R_{\text{goal}}(G)$ [2,1,4]. الهدف هو تحديد المعالج والتردد المناسب لجميع نسخ المهام لتقليل الطاقة المستهلكة [4,5]:

$$E_{\text{total}}(G) = E_{\mathrm{d}}(G) + E_{\mathrm{s}}(G) \qquad (13.1)$$

والتأكد من أن هدف الموثوقية محقق [4,5] :

$$R(G) = \prod_{n_i \in N} R(n_i) \geqslant R_{\text{goal}}(G) \qquad (13.2)$$

5-5 تحديد أولويات المهام (Prioritizing Tasks):

في قسم توصيف المشكلة، تكون الخطوة الأولى هي تحديد أولويات المهام. مشكلة تحديد أولويات المهام مشكلة مهمة لجدولة التطبيق المتوازي المعتمد على مخطط DAG في أنظمة المعالجات المتعددة غير المتجانسة. لا يوجد ما يثبت أن طريقة من طرق تحديد أولويات مهام أفضل من باقي الطرق [1,2,4,5]. ترتيب المهام حسب طريقة قيمة الترتيب التصاعدي

(Ranku) (المعادلة (14)) تعتبر من أكثر الطرق شيوعاً في تحديد أولويات المهام المترابطة في التطبيقات المتوازية المعتمدة على مخطط DAG في أنظمة المعالجات المتعددة غير المتجانسة، وقد تم استخدامها في كثير من الدراسات منها [4]. لذلك سنستخدم طريقة قيمة الترتيب التصاعدي في هذه الدراسة أيضاً. وبالتالي، سيتم ترتيب المهام بترتيب تنازلي لقيمة Ranku، والتي يتم الحصول عليها باستخدام المعادلة التالية:

$$\mathrm{rank_u}(n_i) = \bar{w_i} + \max_{n_j \in \mathrm{succ}(n_i)} \{c_{i,j} + \mathrm{rank_u}(n_j)\} \qquad (14)$$

حيث $\bar{w_i}$ هو متوسط زمن التنفيذ الأسوأ WCET للمهمة n_i باستخدام الترددات القصوى للمعالجات وتحسب كالتالي:

$$\bar{w_i} = \left(\sum_{k=1}^{|U|} w_{i,k}\right) / |U| \qquad (15)$$

يوضح الجدول (2) قيم الترتيب التصاعدي لجميع المهام في الشكل (2) . إذا تم إسناد جميع المهام السابقة المباشرة للمهمة n_i ، عندئذ يمكن إسناد المهمة n_i للمعالج. بفرض لدينا مهمتين n_i و n_j وكان $\mathrm{rank_u}(n_i) > \mathrm{rank_u}(n_j)$. إذا لم يكن هناك قيد أسبقية موجود بين n_i و n_j فإنه من غير الضروري أن تأخذ المهمة n_i الأسبقية في الإسناد والتخصيص. لذلك، فإن ترتيب الإسناد و التخصيص لمهام التطبيق G هو $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

الجدول (2) قيمة Ranku لمهام التطبيق G في الشكل (2) [4].

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$\mathrm{rank_u}(n_i)$	108	77	80	80	69	63.3	42.7	35.7	44.3	14.7

6- الدراسة المرجعية :

عدد قليل من الأبحاث أهتم بمناقشة مشكلة جدولة مهام الزمن الحقيقي في المعالجات المتعددة غير المتجانسة بهدف توفير الطاقة مع مراعاة قيد الموثوقية. منها الخوارزمية التي

قدمها الباحث Xie في [4] وذلك بهدف جدولة مجموعة من المهام مرتبطة مع بعضها بعلاقة الأسبقية، لكن الخوارزمية المقدمة أهملت القيود الزمنية المفروضة على التطبيق المتوازي. الباحث Pop في الورقة البحثية [18] عنون مشكلة الجدولة في انظمة الزمن الحقيقي مع توفير الطاقة وقيد الموثوقية من خلال تكييف خوارزمية RAPM المقترحة في [20] لتتناسب مع منصات المعالجات المتعددة غير المتجانسة، باستخدام مجموعة من المهام المستقلة. أيضاً Guo في [19] قام بالتعديل على خوارزمية RAPM ،بهدف جدولة التطبيقات المتوازية المؤلفة من مجموعة من المهام المرتبطة ببعضها بعلاقة الموثوقية، من أجل توفير الطاقة المستهلكة وتحقيق قيد الموثوقية المطلوبة من التطبيق، كما أن الخوارزمية المقدمة تستطيع جدولة المهام بطريقة استباقية Offline أو آنية Online. تحقيق الموثوقية في الخوارزمية المقترحة يتم من خلال أسلوب يعتمد على الاسترداد واستغلال مساحات الخمول لدى المعالجات، هذا الأسلوب يعاني من مشكلة في حال حدوث خطأ النقل في وقت متأخر من زمن تنفيذ البرنامج وعدم وجود فراغ لدى المعالجات ليتم اسناده للمهمة المستردة، هذا حتماً سيؤدي إلى فشل في تنفيذ التطبيق. بالاعتماد على [22] قدم الباحث Zhang في [23] خوارزمية لجدولة المهام الصارمة المرتبطة مع بعضها بعلاقة الأسبقية في أنظمة المعالجات المتعددة غير الصارمة بهدف توفير الطاقة مع مراعاة قيد الموثوقية ، الموثوقية المطلوبة من التطبيق يتم تحقيقها من خلال استخدام تقنية الاسترداد وتحديداً كتلة الاسترداد والتي يجب أن تكون موجودة على كل معالج، وحجمها هو زمن تنفيذ المهمة الأكبر من بين المهام المسندة لهذا المعالج وبالتالي تختلف من معالج لأخر، الخوارزمية المقترحة تعتبر مثالية في حال كان هناك متسع كبير من الوقت ما بين الموعد النهائي وما بين زمن تنفيذ أخر مهمة على المعالج، ناهيك عن أن لا يوجد أي اثبات علمي أنه في حال حدوث خطأ نقل على معالج ما أثناء تنفيذ مهمة ما فإنه لن يحدث خطأ نقل أيضاً أثناء استرداد هذه المهمة (أعادة تنفيذها) سواءً على نفس المعالج أو على معالج آخر. أيضاً باستخدام فراغات الخمول بعد اسناد المهام المرتبطة ببعضها إلى المعالجات المتعددة غير المتجانسة قدم الباحثون في [24]

خوارزمية تحاول توفير الطاقة الديناميكية المستهلكة مع تحقيق القيد الزمني، لكن بشكل مميز عن جميع الخوارزميات السابقة تحاول الخوارزمية المقدمة تخفيض درجة حرارة المعالجات من خلال طريقة اسناد المهام وليس من خلال التقنيات المادية (العتاد الصلب). أما الورقة البحثية [25] تعتبر أول مقالة تقدم خوارزمية لجدولة مهام الزمن الحقيقي وتحاول تخفيض الطاقة المستهلكة الديناميكية والطاقة الساكنة سوياً مع مراعاة قيد الموثوقية، لكنها تتعامل مع مهام مستقلة وأنظمة أحادية المعالج.

بشكل عام الأبحاث المقدمة في مجال جدولة مهام نظام الزمن الحقيقي الصارم بالاعتماد على الموثوقية مع مراعاة القيد الزمني المفروض على التطبيق المتوازي من النوع DAG، في بيئة المعالجات المتعددة غير المتجانسة تنقسم إلى نوعين أساسين:

- في النوع الأول لا يتم ذكر القيد الزمني بشكل واضح وصريح، إنما يتم ذكره بشكل ضمني. وعلى الرغم من ذكر الموعد النهائي للتطبيق بشكل ضمني فإن الباحثون في هذا الدراسات يفترضون أن التطبيق يملك وقت كاف ليتم تنفيذه حتى مع استخدام التسامح مع الأعطال بهدف تحقيق الموثوقية المطلوبة من التطبيق، من أشهر الأمثلة عن هذه الدراسات الأوراق البحثية المقدمة في [4,5,13,14]. هذا النوع هو محور هذا البحث.
- النوع الثاني من الدراسات يقوم الباحثون بذكر القيد الزمني بشكل واضح وصريح، ويتم مراعاته في كل خطوة من خطوات تنفيذ الخوارزمية، في هذه الحالة يفترض الباحثون أن التطبيق قد لا يملك الوقت الكاف ليتم تنفيذه مع تحقيق هدف الموثوقية، وبناءً على ذلك فإن تحقيق الموثوقية المطلوبة من التطبيق أو تخفيض استهلاك الطاقة تتم بشرط تحقيق القيد الزمني، من أشهر الأمثلة عن هذه الدراسات الأوراق البحثية المطروحة في [2]. هذا النوع سيتم تناوله بشكل مفصل في بحث مستقل.

6-1 خوارزمية ESRG (Energy-Efficient Scheduling With Reliability Goal):

قدمت هذه الخوارزمية من قبل مجموعة من الباحثين في الورقة البحثية [4]. وتهدف هذه الدراسة إلى مناقشة موضوع العلاقة ما بين الموثوقية واستهلاك الطاقة في جدولة مهام نظم الزمن الحقيقي الحرجة لقضية السلامة مع وجود علاقة أسبقية بين المهام في بيئة المعالجات المتعددة غير المتجانسة. حيث تهدف خوارزمية ESRG إلى تخفيض الطاقة الديناميكية المستهلكة قدر الإمكان لكن مع تحقيق متطلبات الموثوقية في التطبيقات المتوازية الحرجة للسلامة والممثلة من خلال مخطط DAG. عملية تخفيض الطاقة الديناميكية المستهلكة لتنفيذ مجموعة مهام التطبيق المتوازي تتم من خلال استخدام تقنية DVFS المعروفة. أما تحقيق الموثوقية فيتم من خلال التسامح مع الأعطال واستخدام تقنية النسخ المتماثل النشط مع عدد متغير من النسخ.

6-1- 1 مبدأ وطريقة عمل خوارزمية ESRG:

الهدف الأساسي من هذه الخوارزمية هو تخفيض كمية الطاقة المستهلكة قدر المستطاع مع الحفاظ على تحقيق متطلبات الموثوقية. يتم ذلك من خلال مرحلتين:

1- المرحلة الأولى هي تحديد أوليات المهام (prioritizing tasks): من اجل ذلك يتم استخدام تقنية تدعى قيمة الترتيب التصاعدي upward rank value والتي قمنا بشرحها مسبقاً.

2- المرحلة الثانية هي اسناد المهام إلى المعالجات غير المتجانسة (allocating tasks): هذه المرحلة يمكن تقسيمها إلى مسألتين فرعيتين:

- المسألة الأولى: تحديد المعالج المناسب لتنفيذ المهمة بحيث يتم تحقيق متطلبات أو هدف الموثوقية.
- المسألة الثانية: تحديد المعالج المناسب لتنفيذ المهمة والذي يحقق متطلبات الموثوقية ويخفض من الطاقة الديناميكية المستهلكة.

6-1-2 تحقيق هدف الموثوقية في خوارزمية ESRG :

بما أن قيمة موثوقية تطبيق ما هي نتاج قيمة موثوقية كل مهام هذا التطبيق، لذلك في البداية يتم حساب الحد الأعلى لموثوقية كل مهمة $R_{\text{up\_goal}}(n_i)$ من خلال إيجاد الجذر النوني لقيمة الموثوقية المطلوبة من التطبيق، كما يلي [3]:

$$R_{\text{up\_goal}}(n_i) = \sqrt[|N|]{R_{\text{goal}}(G)} \qquad (16.1)$$

بالتالي، إذا كانت قيم الموثوقية لجميع المهام تتجاوز قيم الحد الأعلى لكل مهمة من مهام التطبيق، يجب أن تتجاوز قيمة موثوقية التطبيق الناتجة قيمة هدف الموثوقية.

بفرض أن المهمة المراد اسنادها لأحدى المعالجات غير المتجانسة هي $n_{s(y)}$ ، حيث $n_{s(y)}$ تمثل المهمة التي ترتيبها j th ، لدينا $\{n_{s(1)}, n_{s(2)}, \dots, n_{s(y-1)}\}$ مجموعة المهام المسندة للمعالجات، كما لدينا المجموعة $\{n_{s(y+1)}, n_{s(y+2)}, \cdots, n_{s(|N|)}\}$ تمثل مجموعة المهام غير المسندة للمعالجات. من ثم يتم استخدام المعادلة التالية لحساب الموثوقية الجزئية المطلوبة من كل مهمة من مهام التطبيق [4]:

$$R_{\text{goal}}\left(n_{s(y)}\right) = \frac{R_{\text{goal}}(G)}{\prod_{x=1}^{y-1} R\left(n_{s(x)}\right) \times \prod_{z=y+1}^{|N|} R_{\text{up\_goal}}\left(n_{s(z)}\right)} \qquad (16.2)$$

بذلك، يتم نقل هدف الموثوقية للتطبيق لكل مهمة. طالما أن كل مهمة تحقق متطلبات أو هدف الموثوقية المطلوبة منها، فإنه أيضًا يتم تحقيق هدف موثوقية التطبيق ككل [4].

6-1-3 تخفيض استهلاك الطاقة الديناميكية في خوارزمية ERSG:

تستخدم خوارزمية ERSG تقنية DVFS من أجل تخفيض الطاقة الديناميكية المستهلكة لتنفيذ مهام التطبيق، وبما أن الموثوقية الكلية للتطبيق يتم تحويلها لموثوقية جزئية خاصة بكل مهمة، أيضاً عملية تخفيض الطاقة الديناميكية المستهلكة يتم تحويلها لتصبح خاصة بكل مهمة، وفق الاستراتيجية التالية:

لتنفيذ أي مهمة من مهام التطبيق، يتم اولاً حساب الموثوقية المطلوبة من هذه المهمة، ثم يتم المرور على جميع المعالجات والترددات، ويتم حساب قيمة الموثوقية عند كل معالج وكل تردد يدعمه هذا المعالج، فإذا كانت قيمة الموثوقية عند المرور على الثنائية (معالج، تردد) تحقق قيمة الموثوقية المطلوبة يتم اختيار هذه الثنائية، أما اذا لم تكن تحقق هدف الموثوقية يتم الانتقال إلى الثنائية التالية. في حال عدم وجود نسخة واحدة تحقق الموثوقية المطلوبة من المهمة يتم اختيار الثنائية التي تحقق أكبر نسبة موثوقية ممكنة، ثم يعاد الاختيار مرة أخرى وهكذا حتي يتم الوصول إلى الموثوقية المطلوبة [4].

7- خوارزمية (EESARR) المقترحة (Energy-Efficient Scheduling Algorithm with Reliability Requirements) :

الهدف من هذه الخوارزمية هو توفير الطاقة الديناميكية المستهلكة من قبل المهام، أثناء عملية جدولة مهام تطبيقات نظم الزمن الحقيقي الصارمة الحرجة للسلامة المرتبطة مع بعضها البعض بعلاقة الأسبقية والممثلة من خلال مخطط DAG، في بيئة المعالجات المتعددة غير المتجانسة. مع الحفاظ على هدف الموثوقية، أي الوصول إلى هدف الموثوقية المطلوب من التطبيق.

تستخدم خوارزمية EESARR المقترحة تقنية النسخ المتماثل النشط من أجل تحقيق التسامح مع الأعطال بهدف الوصول إلى متطلبات الموثوقية المطلوبة من التطبيق، حيث يتم التعامل مع قضية الموثوقية على أنها شرط من الواجب تحقيقه شأنه شأن شرط القيد الزمني (الموعد النهائي) المفروض في تطبيقات نظم الزمن الحقيقي الصارمة. أيضاً من أجل تحقيق أقصى قدر ممكن من توفير الطاقة الديناميكية المستهلكة من قبل مهام التطبيق المتوازي، فإن خوارزمية EESARR المقترحة تستخدم تقنية DVFS المشهورة، حيث تأمن هذه التقنية ترددات مختلفة وبالتالي اختلاف في كمية الطاقة الديناميكية المستهلكة من أجل نفس المهمة على نفس المعالج عند استخدام ترددات تنفيذ مختلفة، كما ذكرنا سابقاً.

تكمن الفكرة الأساسية لخوارزمية EESARR المقترحة في نقل الموثوقية الكلية المطلوبة من التطبيق إلى موثوقية جزئية خاصة بكل مهمة، فإذا استطاعت كل مهمة من مهام التطبيق المتوازي تحقيق الموثوقية الجزئية المطلوبة منها، فإن التطبيق ككل يستطيع تحقيق الموثوقية الكلية المطلوبة منه. من أجل تحقيق ذلك تقترح خوارزمية EESARR طريقة تجريبه، تقوم على الاختيار الأمثل للنسخ المتماثلة والمعالجات التي تنفذها وأيضاً الترددات المستخدمة في تنفيذها. من خلال هذه الطريقة التجريبية تستطيع خوارزمية EESARR المقترحة الوصول إلى حد الموثوقية المطلوبة مع توفير للطاقة الديناميكية المستهلكة، أي طالما أن التطبيق يحقق هدف الموثوقية المطلوبة منه فإن التطبيق ككل يعتبر موثوق كما ذكرنا سابقاً، بالتالي يجب الاستفادة من تخفيض نسبة الموثوقية التي يحققها التطبيق طالما أنه يحقق هدف الموثوقية من أجل تأمين أكبر قدر ممكن من التوفير في الطاقة الديناميكية المستهلكة.

7-1آلية عمل خوارزمية EESARR:

تنقسم خوارزمية EESARR لمرحلتين أساسيتين وهما:

1- مرحلة نقل الموثوقية الكلية للتطبيق إلى موثوقية جزئية:

تقوم EESARR بنقل الموثوقية الكلية للتطبيق إلى موثوقية جزئية خاصة بكل مهمة، فإذا استطاعت كل مهمة تحقيق الجزء الخاص بها من الموثوقية، فإن الخوارزمية تستطيع تحقيق متطلبات موثوقية التطبيق ككل. ويتم حساب الموثوقية الجزئية الخاصة بكل مهمة من مهام التطبيق بشكل مشابه للطريقة المستخدمة في خوارزمية ESRG. أي من خلال المعادلة التالية Eq. (16.2).

أي أن خوارزمية EESARR يحاول تخفيض الموثوقية الجزئية المطلوبة من كل مهمة، وبالتالي تخفيض عدد النسخ المتماثلة أو تخفيض ترددات النسخ المتماثلة المطلوبة لتحقيق الموثوقية الجزئية الخاصة بكل مهمة، وبالمحصلة تخفيض كمية الطاقة الديناميكية المستهلكة لتنفيذ مهام التطبيق مع تحقيق هدف موثوقية التطبيق.

2- مرحلة اختيار النسخ المتماثلة للمهام:

الهدف الأساسي من هذه المرحلة هو تحديد المهام التي سيتم تكرارها وعدد النسخ المتماثلة المطلوب تنفيذها لكل مهمة، وترددات هذه النسخ، والمعالجات التي ستقوم بتنفيذ هذه النسخ، ويتم ذلك وفقاً للخطوات التالية:

1- من أجل كل مهمة متاحة من مهام التطبيق n_i يتم المرور على كل معالج متاح $u_{pr(n_i^x)}$ وعلى كل تردد ممكن $f_{pr(n_i^x),hz(n_i^x)}$ ويتم حساب قيم موثوقية النسخة المنفذة $R(n_i^x)$ من المهمة n_i. يتم تخزين هذه القيم ضمن مصفوفة خاصة بكل مهمة تدعى مصفوفة "موثوقية النسخ المتماثلة" للمهمة المحددة، أبعاد هذه المصفوفة هي عدد المعالجات وعدد الترددات التي يدعمها كل معالج.

2- يتم المرور على جميع عناصر مصفوفة موثوقية النسخ المتماثلة الخاصة بالمهمة المحددة، ويتم اختيار النسخة المتماثلة الوحيدة n_i^x التي تملك أصغر كمية طاقة ديناميكية مستهلكة (بغض النظر عن التردد الذي تستخدمه طالما أن المعالج متاح) وقيمة موثوقيتها أكبر من أو تساوي قيمة الموثوقية الجزئية المطلوبة من المهمة المحددة $R(n_i^x) \geq R_{goal}(n_i)$ (في حال تواجدها) و يتم الانتقال للخطوة (3).

3- في حال وجود أو عدم وجود نسخة واحدة قادرة على تحقيق شرط الموثوقية الجزئية المطلوبة من المهمة يتم اختيار نسختين متماثلتين على معالجين متاحين، بحيث تكون قيمة الموثوقية التي تحققاها هاتين النسختين أكبر من أو تساوي قيمة الموثوقية المطلوبة من المهمة، وتملكان أصغر كمية طاقة ديناميكية مستهلكة. وفي حال وجود نسخة وحيدة تحقق شرط موثوقية المهمة، يجب أن يكون مجموع كميتي الطاقة الديناميكية المستهلكة من قبلهما أصغر من كمية الطاقة الديناميكية المستهلكة من قبل النسخة الوحيدة. أي بمعنى أوضح يتم المرور على جميع المعالجات المتاحة وجميع الترددات الممكنة واختيار النسختين اللتين تملكان أصغر كمية طاقة ديناميكية مستهلكة ومجموع كميتي الطاقة الديناميكية المستهلكة أقل من كمية الطاقة الديناميكية

المستهلكة من قبل النسخة الوحيدة المختارة من الخطوة (2)، وقيمة الموثوقية الناتجة عنهما أكبر من أو تساوي قيمة الموثوقية الجزئية المطلوبة من المهمة. في حال عدم وجود مثل هاتين النسختين يتم الانتقال للخطوة (4).

4- في حال عدم إيجاد نسخة وحيدة كما هو مطروح في الخطوة (2)، أو عدم وجود نسخين كما هو مطروح في الخطوة (3)، يتم اختيار النسخة التي تحقق أكبر قيمة موثوقية ممكنة، وتكرر الخطوة (3) وهكذا حتى يتحقق شرط الموثوقية الجزئية الخاص بالمهمة المحددة.

5- عندما تنتهي جميع المهام من تحديد عدد النسخ المتماثلة لكل مهمة وترددتها والمعالجات التي سيتم اسناد النسخ لها، تقوم الخوارزمية بحساب العدد الكلي للنسخ المطلوبة $NR(G)$ ، موثوقية التطبيق ككل $R(G)$، الطاقة الديناميكية المستهلكة من قبل جميع مهام التطبيق $E_d(G)$.

7-2 ترميز خوارزمية EESARR المقترحة:

سنعرض فيما يلي ترميز خوارزمية EESARR المقترحة:

The EESARR Algorithm

Input: $G = (N, W, M, C), U, R_{\text{goal}}(G)$
Output: $R(G), E_{\text{total}}(G)$ and its related values

1: Sort the tasks in a list *PriorityList* by descending order of rank_u values.
2: while (there are tasks in *PriorityList*) do
3: $n_i \leftarrow$ *PriorityList*.out()
4: Calculate $R_{\text{goal}}(n_i)$ using Eq. (16.2)
5: while $\left(R(n_i) < R_{\text{goal}}(n_i)\right)$ do
6: for (each processor $u_k \in U$ and u_k is available) do
7: for (each frequency $f_{k,v}$ in from $f_{k,\text{low}}$ and $f_{k,max}$) do
8: Calculate $R\left(n_i^x, u_{pr(n_i^x)}, f_{pr(n_i^x),hz(n_i^x)}\right)$ for the task n_i and save it in n_i *replica matrix*.
chose the one replica with minimum dynamic energy and reach reliability Goal.

9: if $\left(R\left(n_i^x, u_{pr(n_i^x)}, f_{pr(n_i^x),hz(n_i^x)}\right) \geqslant R_{\text{goal}}(n_i)\right)$ then

10: $R(n_i) \leftarrow R\left(n_i^x, u_{pr(n_i^x)}, f_{pr(n_i^x),hz(n_i^x)}\right)$

11: $E_{\text{d}}(n_i) \leftarrow E_{\text{d}}\left(n_i^x, u_{pr(n_i^x)}, f_{pr(n_i^x),hz(n_i^x)}\right)$ using Eq. (11).

12: if ($E_{\text{d}}(n_i) < min\_replica\_dynamic\_energy$) then

13: $min\_replica\_dynamic\_energy = E_{\text{d}}(n_i)$

14: end if

15: end if

chose the tow replica with minimum dynamic energy and reach reliability Goal if the tow replica have minimum dynamic energy less than the one Replica.

16: if $\left(R\left(n_i^{x1+x2}, u_{pr(n_i^{x1+x2})}, f_{pr(n_i^{x1+x2}),hz(n_i^{x1+x2})}\right) \geqslant R_{\text{goal}}(n_i)\right)$ then

17: if ($E_{\text{d}}(n_i^{x1}) + E_{\text{d}}(n_i^{x2}) < min\_replica\_dynamic\_energy$) then

18: $min\_replica\_dynamic\_energy = E_{\text{d}}(n_i^{x1}) + E_{\text{d}}(n_i^{x2})$

19: $R(n_i) = R\left(n_i^{x1+x2}, u_{pr(n_i^{x1+x2})}, f_{pr(n_i^{x1+x2}),hz(n_i^{x1+x2})}\right)$

20: $E_{\text{d}}(n_i) \leftarrow E_{\text{d}}(n_i^{x1}) + E_{\text{d}}(n_i^{x2})$

21: end if

22: else

23: chose the one replica with maximum reliability value and set u_k is not available

24: end if

25: end for

26: end for

27: end while

28: end while

29: Calculate $R(G)$ using Eq. (10)

30: Calculate $E_{\text{d}}(G)$ using Eq. (12)

31: Calculate $NR(G)$

وفيما يلي شرح لأهم تفاصيل ترميز خوارزمية EESARR المقترحة:

1- في السطر الأول، تقوم خوارزمية EESARR بترتيب مهام التطبيق بترتيب تنازلي لقيم ال rank_{u}، وتخزينها في قائمة أو شعاع يدعى " PriorityList ".

2- في السطور 2-28، تختار خوارزمية EESARR النسخة المتماثلة بشكل متكرر والمعالج المتاح بهدف تحقيق متطلبات موثوقية التطبيق. على وجه التحديد، يتم إجراء التفاصيل التالية:

1) الأسطر 2–4: يتم التحقق من وجود مهام ضمن قائمة PriorityList التي تحوي مهام التطبيق والمرتبة حسب الأولوية، في حال وجود أي مهمة متبقية ولتكن n_i، يتم حساب قيمة الموثوقية الجزئية المطلوبة من المهمة (n_i) R_{goal} من خلال المعادلة (16.2) .Eq.
2) السطر 5: تتم مقارنة قيمة موثوقية التطبيق الفعلية مع قيمة شرط موثوقية التطبيق.
3) الأسطر 6–8: من أجل كل مهمة n_i يتم المرور على جميع المعالجات المتاحة، ومن أجل كل معالج يتم المرور على جميع الترددات التي يدعمها، ويتم حساب موثوقية المهمة على كل معالج وكل تردد ممكن ويتم تخزين قيمة الموثوقية ضمن مصفوفة خاصة بكل مهمة تدعى replica matrix
4) الأسطر 9–15: يتم اختيار النسخة التي تملك أقل طاقة ديناميكية مستهلكة، وتحقق شرط الموثوقية الخاصة بالتطبيق.
5) الأسطر 16–21: يتم المرور على عناصر المصفوفة replica matrix، ويتم اختيار النسختين اللتين تملكان أقل طاقة ديناميكية مستهلكة، وموثوقية هاتين النسختين أكبر من أو تساوي شرط موثوقية المهمة، ومجموع الطاقة الديناميكية المستهلكة من قبل هاتين النسختين أقل من كمية الطاقة الديناميكية المستهلكة من قبل النسخة الوحيدة المختارة (في حال تواجدها).
6) الأسطر 22–23: في حال عدم وجود نسخة وحيدة أو نسختين تحققان شرط موثوقية المهمة، يتم اختيار النسخة التي تحقق أكبر قيمة موثوقية ممكنة، وتكرر الخطوات السابقة.

3- الأسطر 29–31، تقوم خوارزمية EESARR بحساب موثوقية التطبيق ككل، الطاقة الديناميكية الكلية المستهلكة من قبل مهام التطبيق جميعها، والعدد النهائي لنسخ التطبيق المتماثلة بالترتيب.

8- نموذج المخطط المتوازي المستخدم في المحاكاة

استخدمنا في هذا البحث برنامج Matlab 2015Rb 64bit الغني عن التعريف كبيئة لمحاكاة عمل الخوارزميات المقترحة وتحليل نتائجها، وبهدف تقييم أداء خوارزمية جدولة مهام نظم الزمن الحقيقي الصارمة المقترحة في هذا البحث، سنستخدم ثلاثة نماذج أساسية من

المخططات المتوازية التي ترتبط فيها المهام بعلاقة الأسبقية و يتم تمثيلها من خلال مخطط DAG. وهذه النماذج هي:

1- نموذج التطبيق البسيط "Sample Application Model":

يتألف هذا النموذج من عشر مهام فقط ترتبط هذه المهام مع بعضها البعض من خلال علاقة الأسبقية، المهمة الأولى من هذه المهام هي مهمة الدخل، أما المهمة العاشرة هي مهمة الخرج، يتم تنفيذه هذه المهام على ثلاثة معالجات غير متجانسة. وهو نفس النموذج الممثل في الشكل 2 والذي قمنا بشرحه في الفقرات السابقة.

2- نموذج تطبيق مركبات متحركة حقيقي "Real-life Automotive Model Application":

تم طرح نموذج المركبات المتحركة هذا في أطروحة الدكتوراه [16] المقدمة من قبل J. Gan, P. Pop, and J. Madsen عام 2014. كما تم استخدامه في العديد من الأوراق البحثية منها [3].

يتألف نموذج تطبيق المركبات المتحركة هذا من ستة كتل وظيفية كما هو واضح في الشكل 3 وهذه الكتل هي [3]:

1- التحكم بالمحرك (Engine Controller): تتألف هذه الكتلة من سبعة مهام $(n_1 - n_7)$.

2- صندوق تغيير السرعة (Automatic Gear Box): مؤلفة من أربعة مهام $(n_8 - n_{11})$.

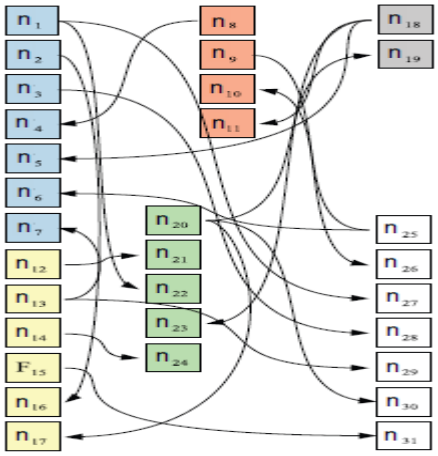
3- نظام الفرامل (Anti-Locking Brake System): وتتألف من ستة مهام $(n_{12} - n_{17})$.

4- مستشعر زاوية العجلة (Wheel Angle Sensor): تتألف من مهمتين فقط $(n_{18} - n_{19})$.

5- التحكم بالتوقف عن العمل (Suspension Controller): تتألف هذه الكتلة من خمسة مهام $(n_{20} - n_{24})$.

6- كتلة عمل جسم المركبة المتحركة (Body Work): وتتألف من سبعة مهام $(n_{25} - n_{31})$.

تظهر كل كتلة من كتل التطبيق الستة في الشكل 3 بلون مختلف، كما يظهر في الشكل علاقة الأسبقية ما بين مهام التطبيق. علماً أن المهام يتم تنفيذها على ثلاثة معالجات غير متجانسة وبالتالي زمن تنفيذ المهمة يختلف من معالج لأخر وهي قيم عشوائية تتبع التوزيع الاحتمالي ضمن المجال $100ms \leqslant w_{i,k} \leqslant 400ms$، كما أن كلفة نقل الرسالة من مهمة سابقة إلى مهمة لاحقة تم إسنادها لمعالج أخر هي أيضاً قيمة عشوائية تتبع التوزيع الاحتمالي وضمن المجال $100ms \leqslant c_{i,j} \leqslant 400ms$ [4].



الشكل 3 نموذج تطبيق مركبات متحركة حقيقي [4].

3- نموذج التطبيقات المتوازية المولدة عشوائيًا " Randomly Generated Parallel Applications Model":

من أجل اختبار قدرات وفوائد خوارزميات جدولة مهام نظم الزمن الحقيقي الصارمة المقترحة على نطاق واسع، قمنا باختبار هذه الخوارزميات على التطبيقات المتوازية الممثلة بمخطط DAG، والتي يتم إنشاؤها عشوائيًا بواسطة برنامج TGFF (Task For Free) 3.6 [17] Graphs. أما عدد العقد (المهام) في المخطط المولد عشوائياً، وبشكل مشابه للورقة البحثية [3]،

سنقوم بتوليد مخططات عشوائية مؤلفة من تدرج من عدد العقد يبدأ هذه التدرج من 50 عقدة (حجم صغير) وينتهي ب 500 عقدة (حجم كبير) وبمقدار زيادة يبلغ 50 عقدة، وهذه العقد ترتبط ببعضها بعلاقة الأسبقية، بحيث يتم إظهار أداء الخوارزميات بشكل واضح عند تطبيقها على المخططات المولدة عشوائياً.

9- المقاييس التجريبية وقيم المعاملات:

تم اختيار مقياسين لمقارنة الخوارزميتين المدروستين في هذه الدراسة بهدف اختبار قدرة وكفاءة هاتين الخوارزميتين، وتحديد إمكانية وقدرة الخوارزمية المقترحة EESARR على تحقيق الهدف المرجو منها . وهذه المقاييس هي:

1- موثوقية التطبيق:

الموثوقية المطلوب من التطبيق ليس عامل تحسن بقدر ما هي شرط لازم في تطبيقات نظم الزمن الحقيقي الصارمة من النوع الحساس لقضية السلامة، حيث أن فشل الجدولة في هذا النوع من التطبيقات قد يؤدي لنتائج كارثية. بالتالي فإن هدف الخوارزمية المقترحة في هذا البحث ليست تحسين الموثوقية بقدر ما هو تحقيق شرط أو هدف موثوقية التطبيق. في هذا البحث سنقوم بالمقارنة ما بين الخوارزميتين من أجل متطلبات موثوقية (هدف الموثوقية) تتراوح من 0.9 حتى 0.99 وبمعدل زيادة يبلغ 0.01، أي من 90% إلى 99%، وذلك من أجل نموذج التطبيق البسيط و نموذج تطبيق المركبات المتحركة الحقيقي، كما في الدراسات [7,15,2,4]، أما من أجل نموذج التطبيقات المتوازية المولدة عشوائيًا فسنقوم بمقارنة الخوارزميات المدروسة من أجل ثلاثة قيم لهدف الموثوقية، وهي أما موثوقية منخفضة تبلغ 0.9 (أي 90%)، موثوقية متوسطة تبلغ 0.95 (أي 95%)، وموثوقية عالية تبلغ 0.99 (أي 99%)[3]. أما بالنسبة لثابت معدل الأخطاء لكل معالج فهو قيمة عشوائية ضمن المجال $0.0010 \leq \lambda_k \leq 0.0020$ ويختلف من معالج لأخر بسبب الطبيعة غير المتجانسة للمعالجات [4].

2- الطاقة الديناميكية المستهلكة:

يقصد بها الطاقة التي تستهلكها مهام التطبيق أثناء عملية التنفيذ على المعالجات المتعددة وغير المتجانسة.

أما بالنسبة لمعاملات الطاقة العامة (المشتركة ما بين نماذج التطبيقات الثلاثة المقترحة) التي سيتم استخدامها في المقارنة فهي موضحة في الجدول التالي:

الجدول (3) معاملات الطاقة المستخدمة في التجارب [4].

الرمز	الشرح	القيمة
C_{ef}	السعة أو القدرة التبديلية الفعالة للمعالج أي عند انتقاله من تردد لأخر.	قيمة عشوائية تتراوح ضمن المجال $0.5 \leq C_{\mathrm{k,ef}} \leq 1.0$
m_{k}	أس القوة الديناميكية، الذي يجب ألا يقل عن 2.	قيمة عشوائية تتراوح ضمن المجال $2.00 \leq m_{\mathrm{k}} \leq 3.00$
P_{ind}	الطاقة الديناميكية المستقلة عن التردد.	قيمة ثابتة لكل المعالجات وتبلغ 0.005
f_{min}	أصغر تردد يمكن أن يدعمه المعالج.	قيمة عشوائية تتراوح ضمن المجال $0.25 \leq f_{min} \leq 0.30$
f_{max}	أكبر تردد يمكن أن يدعمه المعالج.	قيمة ثابتة لكل المعالجات وتبلغ 1

هذا البحث يهتم فقط بمرحلة التصميم، وبالتالي جميع القيم التي قمنا بذكرها ضمن هذه الفقرة هي قيم معروفة خلال مرحلة التحليل، وتم تبنيها في العديد من الأوراق البحثية والمقالات العلمية المحكمة والمنشورة عالمياً.

10- الاختبارات:

تهدف الاختبارات في هذا البحث إلى مقارنة أداء خوارزمية (Energy-Efficient Scheduling With Reliability Goal) ESRG وخوارزمية EESARR (Energy-Efficient Scheduling Algorithm with Reliability Requirements) من حيث الموثوقية المحققة والطاقة المستهلكة من قبل مهام التطبيق.

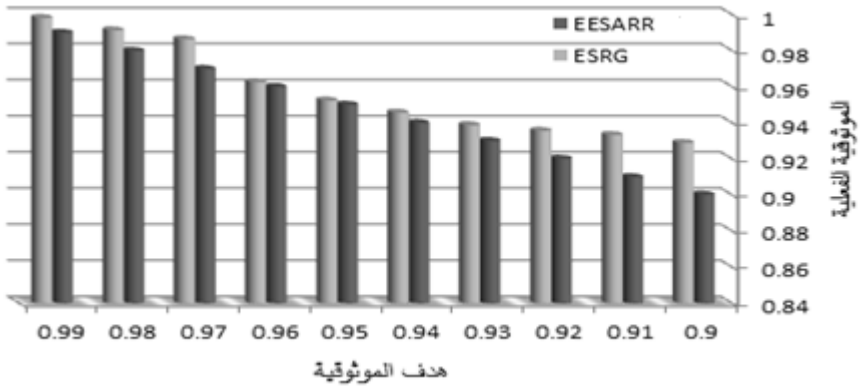
10-1 توصيف وتحقيق الاختبار الأول:

يمكن تلخيص معاملات الاختبار من خلال الجدول التالي :

الجدول (4) معاملات الاختبار الأول.

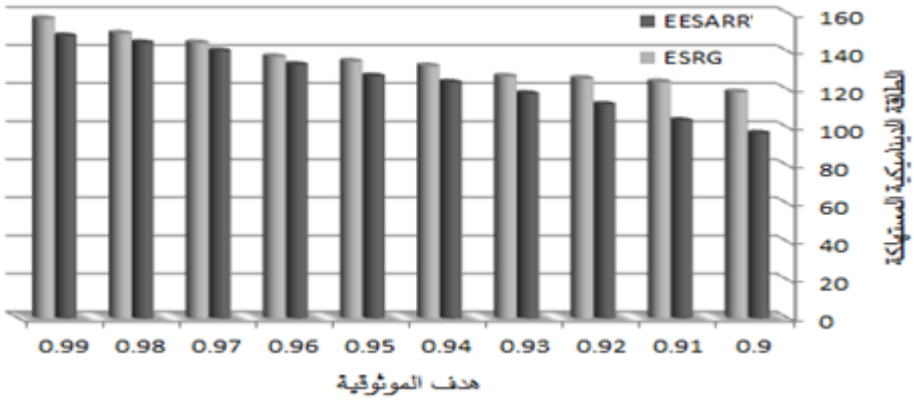
النموذج المستخدم	نموذج التطبيق البسيط
عدد العمليات	10 عمليات
عدد المعالجات	3 معالجات
متطلبات الموثوقية	من 0.9 حتى 0.99 بمعدل زيادة يبلغ 0.01
ثابت معدل أخطاء المعالج	$0.0010 \leq \lambda_k \leq 0.0020$
زمن تنفيذ العملية	$5\ \text{s} \leqslant w_{i,k} \leqslant 21\ \text{s}$
زمن نقل الرسالة من عملية لأخرى	$9\ \text{s} \leqslant c_{i,j} \leqslant 27\ \text{s}$
معاملات الطاقة المستخدمة	من الجدول 2
عدد مرات تكرار التجربة	10

نتائج هذا الاختبار تظهر من خلال المخططين التاليين، حيث يظهر في الشكل (4) موثوقية التطبيق في الخوارزميتين المذكورتين، ومن خلال هذا المخطط نستطيع أن نرى بوضوح أن قيمة الموثوقية الفعلية في خوارزمية ESRG أكبر من قيمة الموثوقية الفعلية المحققة بواسطة خوارزمية EESARR، على أية حال فإن خوارزمية EESARR تحقق هدف الموثوقية.



الشكل (4) قيمة الموثوقية الفعلية في تجربة النموذج البسيط.

أما الشكل (5) يظهر الطاقة الديناميكية المستهلكة من قبل مهام التطبيق، وكما هو واضح أن كمية الطاقة الديناميكية المستهلكة من قبل المهام عند استخدام خوارزمية ESRG أكبر من قيمة الطاقة الديناميكية المستهلكة عند استخدام خوارزمية EESARR.



الشكل (5) قيمة الطاقة الديناميكية المستهلكة في تجربة النموذج البسيط.

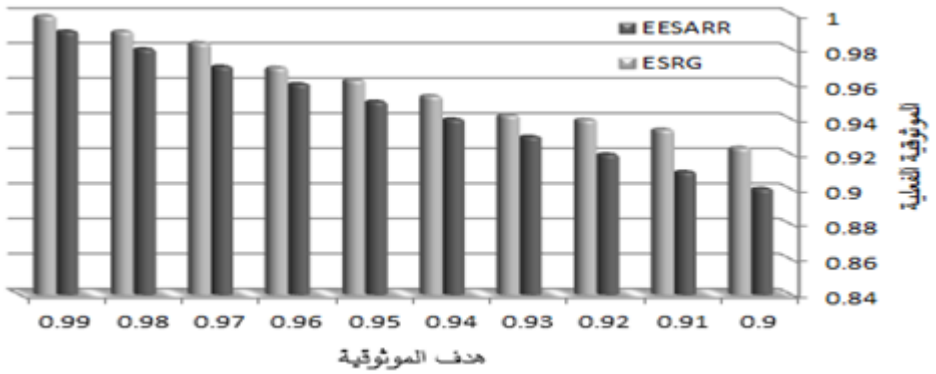
10-2 توصيف وتحقيق الاختبار الثاني:

يمكن تلخيص معاملات الاختبار من خلال الجدول التالي :

الجدول (5) معاملات الاختبار الثاني.

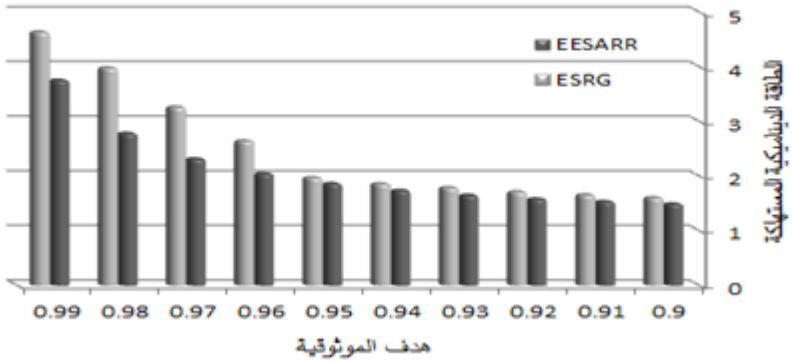
النموذج المستخدم	نموذج تطبيق مركبات متحركة حقيقي
عدد العمليات	31 عملية
عدد المعالجات	3 معالجات
متطلبات الموثوقية	من 0.9 حتى 0.99 بمعدل زيادة يبلغ 0.01
ثابت معدل أخطاء المعالج	$0.0010 \leq \lambda_k \leq 0.0020$
زمن تنفيذ العملية	$100\ ms \leqslant w_{i,k} \leqslant 400\ ms$
زمن نقل الرسالة من عملية لأخرى	$100\ ms \leqslant c_{i,j} \leqslant 400\ ms$
معاملات الطاقة المستخدمة	من الجدول 2
عدد مرات تكرار التجربة	10

نتائج هذا الاختبار تظهر من خلال الشكلين التاليين، حيث يظهر في الشكل (6) موثوقية التطبيق في الخوارزميتين المذكورتين، ومن خلال هذا الشكل، كما في التجربة السابقة نستطيع أن نرى بوضوح أن قيمة الموثوقية الفعلية في خوارزمية ESRG أكبر من قيمة الموثوقية الفعلية المحققة بواسطة خوارزمية EESARR، مع ذلك فإن خوارزمية EESARR تحقق هدف الموثوقية وهو المطلوب.



الشكل (6) قيمة الموثوقية الفعلية في تجربة نموذج تطبيق المركبات المتحركة الحقيقي.

أما الشكل (7) يظهر الطاقة الديناميكية المستهلكة من قبل مهام التطبيق، وكما هو واضح أن كمية الطاقة الديناميكية المستهلكة من قبل المهام عند استخدام خوارزمية EESARR أقل من كمية الطاقة الديناميكية المستهلكة عند استخدام خوارزمية ESRG.



الشكل (7) قيمة الطاقة الديناميكية المستهلكة في تجربة نموذج تطبيق المركبات المتحركة الحقيقي.

10-3 توصيف وتحقيق الاختبار الثالث:

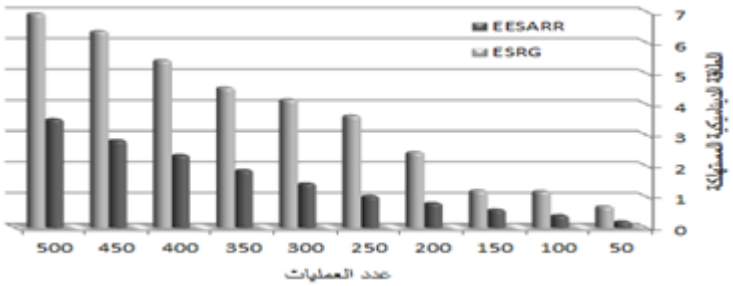
يمكن تلخيص معاملات الاختبار الثالث من القسم الأول من خلال الجدول التالي :

الجدول (6) معاملات الاختبار الثالث.

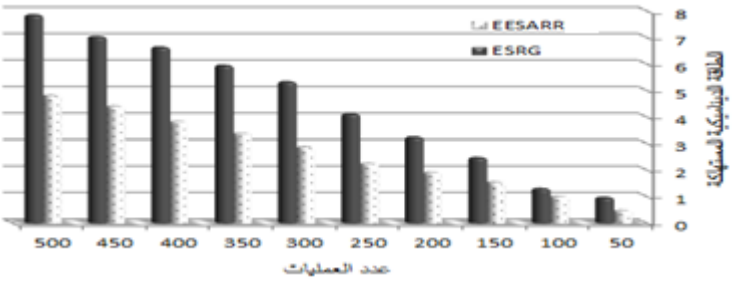
النموذج المستخدم	نموذج التطبيقات المتوازية المولدة عشوائيًا
عدد العمليات	تدرج من 50 حتى 500 عملية بمقدار زيادة 50 عملية
عدد المعالجات	10 معالجات
متطلبات الموثوقية	من 0.9 حتى 0.99 بمعدل زيادة يبلغ 0.05
ثابت معدل أخطاء المعالج	$0.0010 \leq \lambda_k \leq 0.0020$
زمن تنفيذ العملية	$10\ ms \leqslant w_{i,k} \leqslant 50\ ms$
زمن نقل الرسالة من عملية لأخرى	$10\ ms \leqslant c_{i,j} \leqslant 50\ ms$
معاملات الطاقة المستخدمة	من الجدول 2
عدد مرات تكرار التجربة	10

يمكن أن نرى نتائج هذا الاختبار من خلال الأشكال الثلاثة التالية، حيث يظهر في الأشكال 8,9,10 كمية الطاقة الديناميكية المستهلكة عند هدف موثوقية يبلغ 90%، 95%،99% على التوالي.

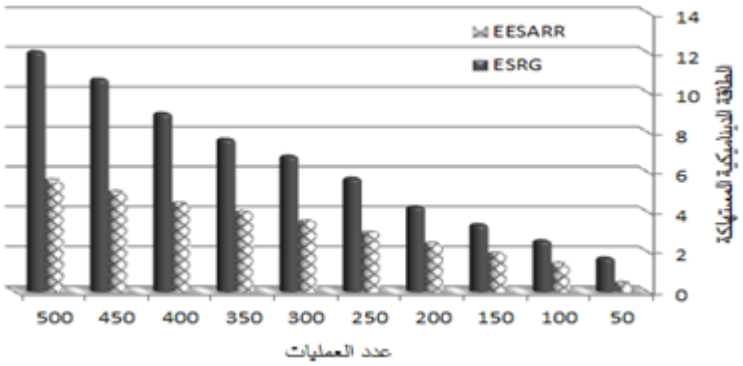
ومن خلال هذا الأشكال الثلاثة، نستنتج أن كمية الطاقة الديناميكية المستهلكة من قبل المهام عند استخدام خوارزمية EESARR أقل من كمية الطاقة الديناميكية المستهلكة عند استخدام خوارزمية ESRG.



الشكل (8) كمية الطاقة الديناميكية المستهلكة في تجربة نموذج التطبيقات المتوازية المولدة عشوائيًا عند هدف موثوقية 90%.



الشكل (9) كمية الطاقة الديناميكية المستهلكة في تجربة نموذج التطبيقات المتوازية المولدة عشوائيًا عند هدف موثوقية 95%.



الشكل (10) كمية الطاقة الديناميكية المستهلكة في تجربة نموذج التطبيقات المتوازية المولدة عشوائيًا عند هدف موثوقية 99%.

11- الخلاصة والنتائج :

من خلال نتائج الاختبارات السابقة يمكننا أن نرى بوضوح أن خوارزمية EESARR استطاعت تحقيق الهدف المرجو منها، وهو تخفيض كمية الطاقة الديناميكية المستهلكة من قبل مهام التطبيق المتوازي الممثل من خلال مخطط DAG، مع تحقيق هدف الموثوقية المطلوبة من التطبيق. ذلك بغض النظر عن نموذج التطبيق المستخدم، أي سواء كان التطبيق عشوائياً، حقيقياً، أو بسيطاً. ويعود السبب في ذلك إلى أن خوارزمية ESRG تعمل على إيجاد نسخة المهمة التي تحقق الموثوقية الجزئية المطلوبة منها والتي تملك أقل تردد ممكن، ونظرياً ذلك سيؤدي إلى انخفاض استهلاك الطاقة، لكن بسبب الطبيعة غير المتجانسة للمعالجات فإن التردد المستخدم على معالج ما قد يستهلك طاقة أقل أو أكثر من معالج أخر عند استخدام نفس التردد، كما أن المهمة نفسها تملك زمن تنفيذ مختلف عند إسنادها على معالج مختلف، وبالتالي فإن خوارزمية EESARR المقترحة تعمل على إيجاد وإسناد نسخة المهمة التي تحقق الموثوقية المطلوبة وتكون الأقل استهلاكاً للطاقة بغض النظر عن التردد المستخدم في تنفيذ هذه المهمة.

أما من ناحية الموثوقية، من خلال التجارب السابقة نستنتج أن خوارزمية ESRG تحقق قيمة موثوقية أعلى من خوارزمية EESARR، وذلك أفضل بالطبع، لكنه مترافق مع زيادة في كمية الطاقة الديناميكية المستهلكة، ويعود السبب في ذلك إلى أن استخدام التردد الأقل الذي

ينفذ المهمة بالموثوقية المطلوبة قد يؤدي إلى اختيار نسخ موثوقيتها أكبر من الموثوقية المطلوبة، لأن خوارزمية ESRG تأخذ النسخة ذات التردد الأقل، بغض النظر عن الموثوقية المحققة طالما هي أكبر من هدف الموثوقية الجزئية للمهمة، أما بالنسبة لخوارزمية EESARR المقترحة تبحث عن النسخة الأقل استهلاكاً للطاقة طالما تحقق الموثوقية المطلوبة، وعملياً تختار النسخة التي تملك أقل طاقة مستهلكة وأقل موثوقية أكبر من هدف الموثوقية الجزئية المطلوبة من المهمة، أي أن زيادة الموثوقية لا تأتي بكلفة مجانية، لكن طالما أن خوارزمية EESARR استطاعت تحقيق هدف الموثوقية المرجو من التطبيق، بالتالي تعتبر عملية الجدولة موثوقة، أي أن خوارزمية EESARR موثوقة وهي أفضل في مجال تخفيض الطاقة الديناميكية المستهلكة، وهو المطلوب.

المراجع:

1- Xiao, X., Xie, G., Li, R., Li, K.: "Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems". In: Proceedings of the 14th IEEE International Symposium on Parallel Distributed Processing with Applications, pp. 1471–1476. IEEE Computer Society (2016).

2- Scheduling Parallel Applications on Heterogeneous Distributed Systems by Guoqi Xie, Gang Zeng, Renfa Li, Keqin Li. Springer ,2019.

3- G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, and K. Li, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," IEEE Trans. Ind. Informat., vol. 13, no. 4, pp. 1629–1640, Aug. 2017.

4- G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, and K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," IEEE Transactions on Sustainable Computing, vol. PP, no. 99, pp. 1–1, Jun. 2017.

5- G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications on heterogeneous systems," IEEE Trans. Ind. Informat., vol. PP, pp. 1–1, Mar. 2017.

6- Hermann Kopetz ,Real-Time Systems Design Principles for Distributed Embedded Applications ,Second Edition,2011.

7- Tian Guo, Jing Liu,Wei Hu, and MengxueWei. "Energy-Aware Fault-Tolerant Scheduling Under Reliability and Time Constraints in Heterogeneous Systems". Springer, 36–46. 2018.

8- RELIABILITY OF COMPUTER SYSTEMS AND NETWORKS , Fault Tolerance, Analysis, and Design by MARTIN L. SHOOMAN,2002.

9- B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," ACM Trans. Des. Autom. Electron. Sys, vol. 18, no. 2, pp. 1–21, 2013.

10- Samal, A.K.; Mall, R.; Tripathy, C.: Fault tolerant scheduling of hard real-time tasks on multiprocessor system using a hybrid genetic algorithm. Swarm Evol. Comput. 14, 92–105 (2014).

11- Xiao, X., Xie, G., Li, R., Li, K.: Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems, pp. 1471–1476. IEEE Computer Society (2016).

12- Li, K. Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers. IEEE Trans. Comput. 61(12), 1668–1681 (2012).

13- Laiping Zhao, Yizhi Ren, Yang Xiang, Sakurai, K., "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems", IEEE International Conference on High Performance Computing and Communications (HPCC), PP. 434 – 441, 2010.

14- Zhao, L., Ren, Y., Sakurai, K.: Reliable workflow scheduling with less resource redundancy. Parallel Comput. 39 (10), 567–585 (2013).

15- M. Wei, J. Liu, T. Li, X. Xu, W. Hu, and D. Zhao, “Fault-tolerant scheduling of real-time tasks on heterogeneous systems,” in IEEE Conference on Industrial Electronics and Applications, pp. 1006–1011,2017,.

16- J. Gan, P. Pop, and J. Madsen, “Tradeoff analysis for dependable real-time embedded systems during the early design phases,” Ph.D. dissertation, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Informatics and Mathematical ModelingInstitut for Informatik og Matematisk Modellering, 2014.

17- R. P. Dick, D. L. Rhodes, and W. Wolf, “Tgff: task graphs for free,” in Proc. 6th int. workshop on Hardware/software codesign. IEEE Computer Society, 1998, pp. 97–101.

18- P. Pop, V. Izosimov, and P. Eles, “Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems,” in IEEE/ACM Int. Conf. Hardware/software Code and Syst Synth, 2007, pp. 233–238.

19- Y. Guo, D. Zhu, and H. Aydin, “Reliability-aware power management for parallel real-time applications with precedence constraints,” in Proc. Int. Conf. Green Comput, 2011, pp. 1–8.

20- D. Zhu and H. Aydin, “Reliability-aware energy management for periodic real-time tasks,” IEEE Trans. Comput, vol. 58, no. 10, pp. 1382–1397, 2009.

21- Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "Ead and pebd: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," IEEE Trans. Comput., vol. 60, no. 3, pp. 360– 374, Jan. 2011.

22- and D. Zhu, "Enhanced reliability-aware power management through shared recovery technique," in IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD), 2009, pp. 63-70.

23- L. Zhang, K. Li, K. Li, and Y. Xu, "Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems," Int. J. Electron. Power & Energy Syst, vol. 78, pp. 499-512, 2016.

24- A. Abdi and H. R. Zarandi, "Hystery: a hybrid scheduling and mapping approach to optimize energy consumption and lifetime reliability of heterogeneous multiprocessor systems," Journal of Supercomputing, vol. 74, no. 5, pp. 2213-2238, 2018.

25- M. Lin, Y. Pan, L. Yang, M. Guo, and N. Zheng, "Scheduling co-design for reliability and energy in cyber-physical systems," IEEE Trans. Emerg. Topics. Comput, vol. 1, no. 2, pp. 353-365, 2014.

26- ISO, "Iso 26262-road vehicles-functional safety," International Organization for Standardization in ISO 26262, 2011.