

OPERATING SYSTEM

Lecture Notes

Dr. Professor, J.M. Khalifeh

قسم المعلوماتية

الوحدة الرابعة

النسخة العربية

Unit-4

CPU Scheduling

ملخص

جدولة وحدة المعالجة المركزية هي عملية تسمح لعملية واحدة باستخدام وحدة المعالجة المركزية بينما تتأخر عملية أخرى (في وضع الاستعداد) بسبب عدم توفر أي موارد مثل الإدخال / الإخراج وما إلى ذلك، وبالتالي الاستفادة الكاملة من وحدة المعالجة المركزية. الغرض من جدولة وحدة المعالجة المركزية هو جعل النظام أكثر كفاءة وعدالة وأسرع. تحدد جدولة وحدة المعالجة المركزية الطريقة والترتيب الذي يجب تنفيذ العمليات به. في أنظمة التشغيل الحديثة، تتم جدولة المسالك على مستوى النواة - وليس العمليات - في الواقع بواسطة نظام التشغيل. ومع ذلك، غالبًا ما يتم استخدام المصطلحين "جدولة العملية" و "جدولة المسلك" ليشيران إلى نفس المفهوم في هذه الوحدة وسنستخدم جدولة العملية عند مناقشة مفاهيم الجدولة العامة. سنقدم هنا مفاهيم أساسية لجدولة وحدة المعالجة المركزية وأنواع جدولة وحدة المعالجة المركزية والمعايير التي تأخذها وحدة المعالجة المركزية في الاعتبار أثناء عمليات "الجدولة"، كما نقدم العديد من خوارزميات جدولة وحدة المعالجة المركزية، أما ما يتعلق بجدولة النظم متعددة المعالجات والمتعددة النواة ونظم الزمن الحقيقي فسيتم تأجيل جدولتها إلى نظم التشغيل المتقدمة.

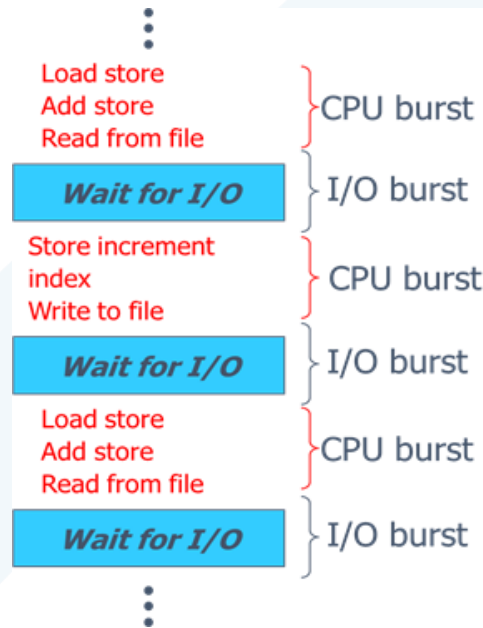
أهداف الوحدة

- وصف خوارزميات جدولة وحدة المعالجة المركزية المختلفة.
- تقييم خوارزميات جدولة وحدة المعالجة المركزية على أساس معايير الجدولة.
- وصف خوارزميات الجدولة المستخدمة في أنظمة التشغيل Windows و Linux و Solaris.
- تقييم خوارزميات جدولة وحدة المعالجة المركزية.
- تصميم البرامج التي تنفذ العديد من خوارزميات جدولة وحدة المعالجة المركزية المختلفة.

مفاهيم أساسية

يتم تنفيذ جميع البرامج تقريبًا على شكل تتالي دورات معينة من العمليات في وحدة المعالجة المركزية وانتظار إدخال/إخراج من نوع ما. يتم في الأنظمة البسيطة تشغيل عملية واحدة، وهنا يهدر الوقت المستغرق في انتظار الإدخال/الإخراج، وتضييع دورات وحدة المعالجة المركزية هذه دون فائدة. أما في أنظمة البرمجة المتعددة، فإن نظام الجدولة يسمح لعملية واحدة باستخدام وحدة المعالجة المركزية بينما تنتظر أخرى الإدخال/الإخراج، وبالتالي الاستفادة الكاملة من دورات وحدة المعالجة المركزية المهدورة. وهنا فإن التحدي يتمثل في جعل النظام العام "فعالاً" و "عادلاً" قدر الإمكان، ويخضع لشروط متفاوتة ومتغيرة في كثير من الأحيان، وحيثما يكون مصطلح "فعالاً" و "عادلاً" مصطلحات تتعلق بطبيعة التطبيقات وتخضع لتغيير سياسات الأولوية.

تتناوب الشكل 1 جميع العمليات تقريبًا بين حالتين يتم في كل منهما تنفيذ دفعة Burst، وذلك في دورة مستمرة، كما هو موضح في الشكل:



الشكل 1: تناوب تشغيل العمليات

دفعة وحدة المعالجة المركزية لإجراء العمليات الحسابية، و دفعة I/O، في انتظار نقل البيانات داخل أو خارج النظام. تختلف دفعات وحدة المعالجة المركزية من عملية إلى عملية ومن برنامج إلى آخر.

جدولة وحدة المعالجة المركزية

جدولة وحدة المعالجة المركزية هي عملية يستخدمها نظام التشغيل لتحديد أي مهمة أو عملية ستستخدمها في وقت محدد. تُعد هذه العملية مهمة لأن وحدة المعالجة المركزية لا يمكنها معالجة سوى مهمة واحدة في كل مرة، ولكن عادةً ما تكون هناك العديد من المهام التي تحتاج إلى معالجة. ومن ضمن أهداف جدولة وحدة المعالجة المركزية:

- زيادة فعالية استخدام وحدة المعالجة المركزية إلى الحد الأعظمي الممكن
- تقليل زمن الاستجابة وانتظار العملية.

عندما تصبح وحدة المعالجة المركزية خاملة، فإن مهمة جدولة وحدة المعالجة المركزية (المعروفة أيضًا باسم جدولة المدى القصير) تتركز على تحديد عملية أخرى من قائمة الانتظار الجاهزة للتشغيل بعد ذلك. وهناك الكثير من الخوارزميات التي يتم تطبيقها لذلك تبعاً لبنية المنظومة ومتطلبات التطبيق. وجميع هذه الخوارزميات يمكن تصنيفها إلى نوعين رئيسيين من طرق الجدولة الاستيلائية والجدولة غير الاستيلائية.

الجدولة الاستيلائية وغير الاستيلائية Preemptive and non-preemptive Scheduling

تتخذ قرارات جدولة وحدة المعالجة المركزية في ظل واحد من أربعة شروط:

1. عندما تنتقل العملية من حالة التشغيل إلى حالة الانتظار، مثل طلب الإدخال/الإخراج أو الانتظار بناءً على الاستدعاء (wait).
2. عندما تنتقل العملية من حالة التشغيل إلى حالة الاستعداد، على سبيل المثال استجابة لمقاطعة.
3. عندما تنتقل العملية من حالة الانتظار إلى حالة الاستعداد، عند اكتمال الإدخال/الإخراج أو العودة من wait() مثلاً.

4. عندما تنتهي العملية.

بالنسبة للشرطين 1 و 4، يجب تحديد عملية جديدة، لا يوجد خيار آخر.

بالنسبة للشرطين 2 و 3، لدينا الخيار إما لمتابعة تشغيل العملية الحالية، أو تحديد عملية أخرى.

إذا كانت الجدولة تتم فقط في ظل الشرطين 1 و 4، فعندها يُقال إن النظام غير استيلائي أو تعاوني. في ظل هذه الظروف، بمجرد أن تبدأ العملية في العمل، فإنها تستمر في العمل، حتى يتم حظرها طواعية أو حتى تنتهي. وإلا يُقال أن النظام استيلائي في غير ذلك من الحالات.

استخدم Windows جدولة غير استيلائية حتى Windows 3.x، وبدأ في استخدام الجدولة الاستيلائية مع Win95. استخدمت أجهزة Mac غير استيلائية قبل OS X، واستيلائية منذ ذلك الحين. لاحظ أن الجدولة الاستيلائية ممكنة فقط على الأجهزة التي تدعم المقاطعة الزمنية.

لاحظ أن الجدولة الاستيلائية يمكن أن تسبب مشاكل عند مشاركة عمليتين للبيانات، لأن عملية واحدة قد تنقطع في منتصف تحديث بنية البيانات المشتركة. وسنعود إلى ذلك في وحدة أخرى من هذا.

يمكن أن يكون الاستيلاء أيضًا مشكلة إذا كانت النواة مشغولة في تنفيذ استدعاء النظام (على سبيل المثال، عند تحديث هياكل بيانات النواة الهامة) عند حدوث الاستيلاء. تتعامل معظم أنظمة UNIX الحديثة مع هذه المشكلة عن

طريق جعل العملية تنتظر حتى يكتمل استدعاء النظام أو يتم حظرها قبل السماح بالإجراءات الاستثنائية لسوء الحظ، فإن هذا الحل يمثل مشكلة لأنظمة الزمن الحقيقي، حيث لا يعود من الممكن ضمان الاستجابة في الوقت المطلوب. تحمي بعض الأقسام الهامة من التعليمات البرمجية نفسها من مشاكل التزامن عن طريق تعطيل المقاطعات قبل الدخول إلى القسم الحرج وإعادة تمكين المقاطعات عند الخروج من القسم. وغني عن القول أنه يجب أن يتم ذلك فقط في حالات نادرة، و فقط على أجزاء قصيرة جدًا من التعليمات البرمجية التي ستنتهي بسرعة، (عادةً ما تكون مجرد تعليمات قليلة عن الجهاز).

الموزع Dispatcher

الموزع هو الوحدة النمطية التي تمنح التحكم في وحدة المعالجة المركزية للعملية المحددة من قبل الجدول. تتضمن هذه الوظيفة:

- تبديل السياق.
- التبديل إلى وضع المستخدم.
- القفز إلى الموقع الصحيح في البرنامج الذي تم تحميله حديثًا.

يجب أن يكون الموزع أسرع ما يمكن، حيث يتم تشغيله على كل تبديل سياق. يُعرف الوقت الذي يستهلكه الموزع بتأخير التوزيع **dispatch latency**.

معايير الجدولة

هناك العديد من المعايير المختلفة التي يجب مراعاتها عند محاولة تحديد خوارزمية الجدولة "الأفضل" لحالة وبيئة معينة، بما في ذلك:

- استخدام وحدة المعالجة المركزية **CPU Utilization**: من الناحية المثالية، ستكون وحدة المعالجة المركزية مشغولة بنسبة 100% من الوقت، بحيث يتم إضاعة 0 دورة من دورات وحدة المعالجة المركزية. أما من الناحية العملية فإن استخدام وحدة المعالجة المركزية في النظام الحقيقي يتراوح بين 40% (محملة بشكل خفيف) إلى 90% (محملة بكثافة).
- الإنتاجية **Throughput**: وهي عدد العمليات المنجزة لكل وحدة زمنية. قد تتراوح من 10 / ثانية إلى 1 / ساعة حسب العمليات المحددة.
- زمن الانجاز **Turnaround time**: هو الزمن المطلوب لإكمال عملية معينة، من بدء التقديم حتى الانتهاء.
- زمن الانتظار **Waiting Time**: وهو مقدار الزمن الذي تقضيه العمليات في قائمة الانتظار الجاهزة في انتظار دورها للحصول على وحدة المعالجة المركزية.
- زمن الاستجابة **Response time**: وهو الزمن الذي يستغرقه برنامج تفاعلي من بداية إصدار الأمر إلى بدء الرد على هذا الأمر.

والمطلوب بشكل عام، تحسين متوسط قيمة أحد المعايير (زيادة استخدام وحدة المعالجة المركزية وإنتاجيتها، وتقليل جميع المعايير الأخرى). ومع ذلك، في بعض الأحيان يريد المرء أن يفعل شيئاً مختلفاً، مثل تقليل وقت الاستجابة الأقصى.

خوارزميات الجدولة

سوف نقوم بشرح الفقرات التالية العديد من استراتيجيات الجدولة الشائعة، آخذين بالاعتبار للتبسيط وجود عدد صغير من العمليات وأن كل عملية مخصصة بدفعة زمنية واحدة لخدمتها من قبل وحدة المعالجة المركزية. من الواضح أن الأنظمة الحقيقية يجب أن تتعامل مع الكثير من العمليات المتزامنة التي تنفذ دورات من تخصيص الدفعات الزمنية التي تتوزع بين المعالج والمدخلات والمخرجات.

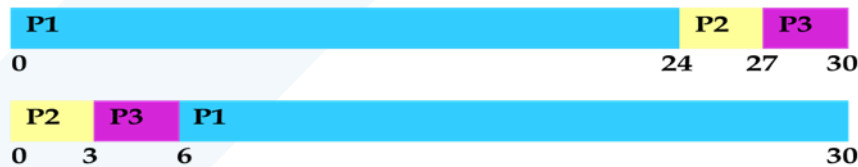
خوارزمية الجدولة الخدمة FCFS

FCFS أي من يأتي أولاً يُخدم أولاً: وهي خوارزمية بسيطة للغاية. يتم فيها تنفيذ العمليات المنتظرة في طابور الانتظار بشكل مشابه لطوابير FIFO، أي مثل الزبائن الذين ينتظرون في طابور في البنك أو مكتب البريد أو في آلة النسخ.

ومع ذلك، لسوء الحظ، يمكن أن ينتج عن FCFS متوسط فترات انتظار طويلة جداً، خاصة إذا كانت العملية الأولى للوصول إلى هناك تستغرق وقتاً طويلاً. على سبيل المثال، ضع في اعتبارك العمليات الثلاث التالية:

Process	Burst Time
P1	24
P2	3
P3	3

في مخطط Gantt الأول أدناه، تصل العملية P1 أولاً فيتم تنفيذها أولاً. متوسط وقت الانتظار للعمليات الثلاث $(27 + 24 + 0) / 3 = 17.0$ ملي ثانية.



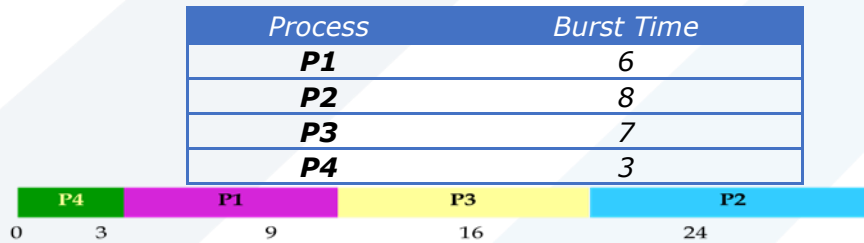
بينما لو غيرنا في الخوارزمية لتنفيذ العملية الأقصر أولاً فإن مخطط Gantt يصبح كما في الحالة الثانية من الشكل ، وسينخفض متوسط زمن الانتظار إلى $(6 + 3 + 0) / 3 = 3.0$ ملي ثانية. إجمالي زمن الانجاز للدفعات الثلاث هو نفسه، ولكن في الحالة الثانية، تنتهي عمليتان من الثلاثة بشكل أسرع، وتتأخر العملية الأخرى بمقدار قصير فقط.

وهذا يبين أنه يمكن لـ FCFS أن يحظر تشغيل عمليات ذات دفعات زمنية صغيرة انتظارا لتنفيذ عملية ذات دفعة زمنية كبيرة لأنها تمتلك أولوية القدوم أولاً. وهذا ما يعرف بتأثير القافلة Convoy Effect. عندما تحظر العمليات ذات الدفعات الكبيرة وحدة المعالجة المركزية، فمثلا يمكن حظر عدد من عمليات الإدخال / الإخراج انتظارا لفرغ مشغولية المعالج، مما يترك أجهزة الإدخال / الإخراج في وضع الخمول. عندما تتخلى العملية عن وحدة المعالجة المركزية أخيراً،

فإن عمليات الإدخال / الإخراج تمر عبر وحدة المعالجة المركزية بسرعة، مما يترك وحدة المعالجة المركزية في وضع الخمول بينما يصطف الجميع في قائمة الانتظار للإدخال / الإخراج، وهكذا في كل طابور.

خوارزمية جدولة المهمة الأقصر زمنياً أولاً، SJF Shortest Job First

تكمن الفكرة وراء خوارزمية SJF في اختيار أسرع مهمة صغيرة يجب القيام بها، وإخراجها من الطريق أولاً، ثم اختيار المهمة التالية الأصغر والأسرع للقيام بها بعد ذلك. (من الناحية الفنية، تختار هذه الخوارزمية عملية ما بناءً على أقصر دفعة زمنية لوحدة المعالجة المركزية التالية، وليس وقت العملية الإجمالي.)
على سبيل المثال، يعتمد مخطط Gantt أدناه على أوقات دفعات وحدة المعالجة المركزية التالية (وافتراض أن جميع الوظائف وصلت في نفس الوقت).



في الحالة أعلاه، يكون متوسط وقت الانتظار $(16 + 9 + 3 + 0) / 4 = 7.0$ مللي ثانية، (مقابل 10.25 مللي ثانية لـ FCFS لنفس العمليات.)

يمكن إثبات أن SJF هو أسرع خوارزمية جدولة، لكنه يعاني من مشكلة واحدة مهمة: كيف تعرف كم من الوقت ستستغرق دفعة وحدة المعالجة المركزية التالية؟

بالنسبة للوظائف المجمعة طويلة الأجل، يمكن القيام بذلك استناداً إلى الحدود التي يضعها المستخدمون لوظائفهم عند إرسالها، مما يشجعهم على وضع حدود منخفضة، ولكنه يخاطر باضطرارهم إلى إعادة إرسال الوظيفة إذا قاموا بتعيين حد منخفض. ومع ذلك، فإن ذلك لا يعمل مع جدولة وحدة المعالجة المركزية قصيرة المدى على نظام تفاعلي.
قد يكون الخيار الآخر هو قياس خصائص وقت التشغيل للوظائف إحصائياً، لا سيما إذا تم تشغيل نفس المهام بشكل متكرر ومتوقع. ولكن مرة أخرى، هذا ليس خياراً قابلاً للتطبيق لجدولة وحدة المعالجة المركزية قصيرة المدى في العالم الحقيقي.

يمكن أن يكون SJF استيلائياً أو غير استيلائياً. يحدث الاستيلاء عندما تصل عملية جديدة إلى قائمة الانتظار الجاهزة التي لها دفعة متوقعة أقصر من الوقت المتبقي في العملية التي تكون دفعتها حالياً في وحدة المعالجة المركزية. يشار إلى SJF الاستيلائياً أحياناً على أنه أقصر وقت متبقي Shortest Remaining Time First بداية الجدولة.

على سبيل المثال، يعتمد مخطط Gantt التالي على البيانات التالية:

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
p4	5	4

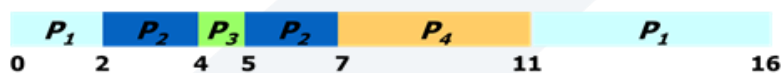
SJF (non-preemptive)



$$\text{Average waiting time} = (0 + 6 + 3 + 7)/4 = 4$$

$$\text{Average Turned around time} = (7 + 4 + 10 + 11)/4 = 8$$

SJF (preemptive)



$$\text{Average waiting time} = (9 + 1 + 0 + 2)/4 = 3$$

$$\text{Average Turned around time} = (16 + 1 + 5 + 6)/4 = 7$$

Preemptive AWT < non Preemptive AWT

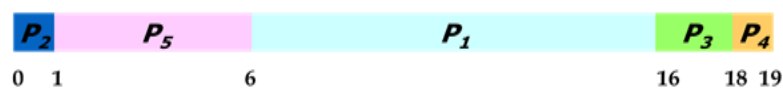
خوارزمية الجدولة القائمة على الأولوية Priority based Scheduling

جدولة الأولوية هي حالة أكثر عمومية لـ SJF، حيث يتم تعيين أولوية لكل وظيفة ويتم جدولة الوظيفة ذات الأولوية القصوى أولاً. (يستخدم SJF معكوس وقت الدفعة المتوقع التالي كأولويته - كلما كانت الدفعة المتوقعة أصغر، زادت الأولوية).

لاحظ أنه من الناحية العملية، يتم تنفيذ الأولويات باستخدام الأعداد الصحيحة ضمن نطاق ثابت، ولكن لا توجد اتفاقية متفق عليها حول ما إذا كانت الأولويات "العالية" تستخدم أعداداً كبيرة أو أعداداً صغيرة. نستخدم هنا عدداً منخفضاً للأولويات العالية، حيث يمثل الصفر أعلى أولوية ممكنة.

على سبيل المثال، يعتمد مخطط Gantt التالي على أوقات دفعات العملية والأولويات المبينة في الجدول، ويؤدي إلى متوسط وقت انتظار يبلغ 8.2 مللي ثانية:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



$$AWT = 8.2$$

$$ATAT = (1 + 6 + 16 + 18 + 19)/5 = 12$$

يمكن تعيين الأولويات إما داخلياً أو خارجياً. يتم تعيين الأولويات الداخلية بواسطة نظام التشغيل باستخدام معايير مثل متوسط وقت الدفعة، ونسبة نشاط وحدة المعالجة المركزية إلى نشاط الإدخال/الإخراج، واستخدام موارد النظام، والعوامل الأخرى المتاحة للنواة. يتم تحديد الأولويات الخارجية من قبل المستخدمين، بناءً على أهمية الوظيفة، والرسوم المدفوعة، والسياسة، وما إلى ذلك.

يمكن أن تكون جدولة الأولوية استيلائية أو غير استيلائية.

يمكن أن تعاني جدولة الأولويات من مشكلة كبيرة تُعرف بالحظر غير المحدود لبعض العمليات، أو التجويع، حيث يمكن أن تنتظر مهمة ذات أولوية منخفضة إلى الأبد نظرًا لوجود دائمًا بعض الوظائف الأخرى التي لها أولوية أعلى.

إذا تم السماح بحدوث هذه المشكلة، فسيتم تشغيل العمليات في النهاية عندما يخف تحميل النظام (في الساعة 2:00 صباحًا على سبيل المثال)، أو ستضيع في النهاية عند إيقاف تشغيل النظام أو تعطله. (وهناك شائعات عن وظائف عالقة منذ سنوات).

أحد الحلول الشائعة لهذه المشكلة هو التقييم العمري للعمليات، حيث تزداد أولويات الوظائف كلما طال انتظارها. بموجب هذا المخطط، ستحصل الوظيفة ذات الأولوية المنخفضة في النهاية على أولويتها مرتفعة بدرجة كافية ليتم تشغيلها.

الأولويات الداخلية والخارجية

تستخدم الأولويات المحددة داخليًا بعض الكميات القابلة للقياس لحساب أولوية العملية. على سبيل المثال:

- حدود الوقت
- متطلبات الذاكرة
- عدد الملفات المفتوحة
- تم استخدام نسبة متوسط دفعات الإدخال / الإخراج إلى متوسط دفعات وحدة المعالجة المركزية في أولويات الحوسبة.
- يتم تحديد الأولويات الخارجية من خلال معايير خارجية لنظام التشغيل، مثل
- أهمية العملية
- الأموال التي يتم دفعها لاستخدام الكمبيوتر
- الجهة الراعية للعمل
- العوامل المتعلقة بسياسات التشغيل

خوارزمية الشريط الدّوار Round robin (RR)

تشبه جدولة الشريط الدّوار جدولة FCFS، باستثناء أنه يتم تعيين دفعات وحدة المعالجة المركزية بحدود تسمى كمية الزمن time quantum.

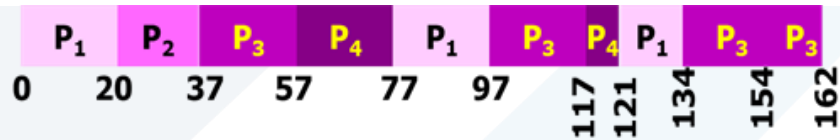
عندما يتم تخصيص وحدة المعالجة المركزية للعملية، يتم تعيين المؤقت بقيمة تم تعيينها لكمية زمنية ما. إذا انتهت دفعة العملية قبل انتهاء فترة المؤقت الكمي، فسيتم إخراجها إلى خارج وحدة المعالجة المركزية تمامًا مثل خوارزمية FCFS العادية.

إذا توقف المؤقت أولاً، فسيتم تبديل العملية من وحدة المعالجة المركزية ونقلها إلى ذيل قائمة الانتظار الجاهزة. يتم الاحتفاظ بقائمة الانتظار الجاهزة كقائمة انتظار دائرية، لذلك عندما يكون لكل العمليات دور، يعطي الجدول العملية الأولى دورًا آخر، وهكذا.

يمكن أن تعطي جدولة RR تأثير مشاركة جميع المعالجات في وحدة المعالجة المركزية بشكل متساوٍ، على الرغم من أن متوسط وقت الانتظار يمكن أن يكون أطول من خوارزميات الجدولة الأخرى. في المثال التالي، يبلغ متوسط وقت الانتظار 5.66 ملي ثانية.

مثال على RR مع كم الوقت = 20

Process	Burst Time
P1	53
P2	17
P3	68
P4	24

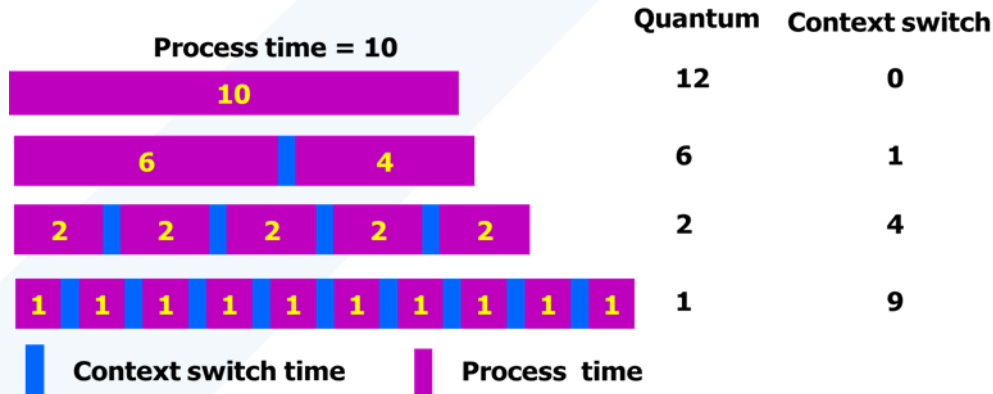


$$ATAT_{(RR)} = (134+37+121+162)/4=113.5$$

$$ATAT_{(SJF)} = (17+41+94+162)/4=78.5$$

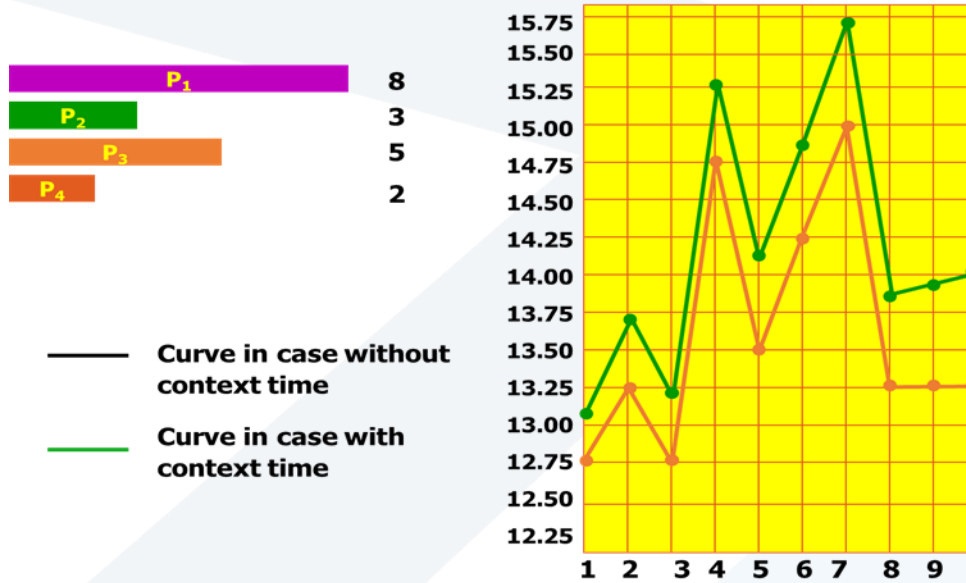
$$ATAT_{(FCFS)} = (53+70+138+162)/4=105.75$$

أداء RR حساس لكم الوقت المختار. إذا كان الكم كبيراً بدرجة كافية، فإن RR يعود إلى خوارزمية FCFS؛ إذا كانت صغيرة جداً، فستحصل كل عملية على $n/1$ من وقت المعالج وتشارك وحدة المعالجة المركزية بالتساوي مع بقية العمليات.



ولكن النظام الحقيقي يستدعي زمناً إضافياً لكل تبديل سياق، وكلما قل كم الوقت، زاد عدد مرات التبديل الموجودة في السياق كما في الشكل أدناه. تستخدم معظم الأنظمة الحديثة قيمة لكم الوقت بين 10 و 100 ملي ثانية، وأوقات تبديل السياق من مرتبة 10 ميكروثانية، وبالتالي فإن الزمن الإضافي الناجم عن تبديل السياق يكون صغيراً بالنسبة إلى كم الوقت.

يختلف وقت الدوران أيضاً باختلاف وقت الكم، بطريقة غير ظاهرة. ضع في اعتبارك، على سبيل المثال، العمليات الموضحة في الشكل أدناه:



بشكل عام، يتم تقليل وقت الاستجابة إلى الحد الأدنى إذا أنهت معظم العمليات دفعة وحدة المعالجة المركزية التالية في غضون كم الوقت المعطى لمرة واحدة. على سبيل المثال، مع ثلاث عمليات دفعات كل منها 10 مللي ثانية، يكون متوسط وقت الاستجابة من أجل وقت كمي مساو لـ 1 مللي ثانية 29، ويقلل الوقت الكمي المساوي لـ 10 مللي ثانية وقت الاستجابة إلى 20. ومع ذلك، إذا كانت القيمة كبيرة جدًا، فإن RR يتدهور إلى FCFS. القاعدة الأساسية هي أن 80% من دفعات وحدة المعالجة المركزية يجب أن تكون أصغر من الوقت الكمي.