الأنظمة الحديثة تتيح الوصول إلى بيانات ملف في القرص (نظام الملفات) بطريقتين:

1. Memory mapping مثل ()mmap

والتي تعالج بواسطة نظام الذاكرة الافتراضية (Virtual Memory subsystem)

2. I/O system calls مثل ()read وَ ()Write

والتي تعالج بواسطة نظام وحدات الإدخال والإخراج.(I/O subsystem)

Disk Cache

آلية لتقليل الوقت المستغرق للقراءة أو الكتابة من وإلى القرص الصلب.

أنواع Disk Cache

1. Buffer Cache

Buffer cache وسيط بين I/O system calls (أوامر القراءة والكتابة) و القرص (نظام الملفات).

في السابق كان يخصص جزء من الذاكرة؛ لـِ Buffer Cache، وحالياً يوجد كجزء من القرص الصلب.

وظيفته:

الاحتفاظ بالبيانات المقروءة حالياً و البيانات المجاورة لها في القرص، في حالة لو طُلبت مستقبلاً. كما يحتفظ بالبيانات المكتوبة حالياً لفترة ثم تنقل إلى القرص.

آلية القراءة:

عند القراءة من القرص، تتحرك ذراع القرص حتى يصل رأس القراءة إلى المسار المطلوب، وبعد فترة زمنية يقوم الرأس بجمع bits. وعادة لا تكون Sectors المقروءة في البداية هي الوحيدة المطلوبة من قبل نظام التشغيل فقد يحتاج إلى زيادة لاحقاً ؛ وبسبب ذلك يتم الاحتفاظ بالـــSectors (Blocks)المقروءة والمجاورة لها في Buffer Cache فتكون جاهزة في حالة طلبها نظام التشغيل لاحقاً.

لماذاBuffer Cache:

لأن سرعة وحدات الإدخال والإخراج أسرع من نقل Bits من وإلى القرص، تستخدم Buffer Cache ليتسنى للاثنين العمل بأقصى سرعة ممكنة دون حدوث تأخير أو إرباك.

2. Page Cache

تنقل بيانات الملف المستخدم والمجاورة لها باستخدام تقنية الذاكرة الافتراضية و تحتفظ بها كـــ Pages بدلاً من Block، أي أنها تتعامل مع العنوان الافتراضي وهذا أسرع من استخدام العنوان الفعلي لـــ blocks .

3. Two cache model

تستخدم Buffer Cache وPage Cache معاً

- في حالة استخدام Buffer Cache (أي عند إصدار I/O system calls)

عند القراءة من ملف تنقل بياناته من القرص إلى الـــ Buffer Cache ومنها إلى التطبيق الذي طلب تلك البيانات.

وفي الكتابة يحدث العكس، حيث تعدل البيانات في Buffer Cache ثم تنقل إلى القرص فيما بعد.

- في حالة استخدام page cache(أي memory mapping I/O)

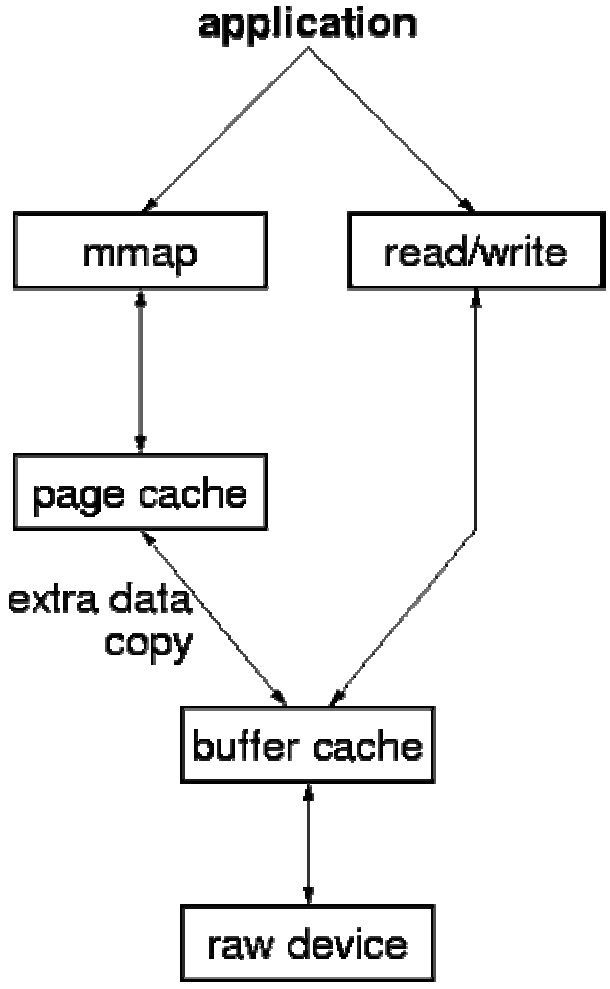
تنقل البيانات من القرص إلى Buffer Cache ولأن الأخير لا يتعامل مع نظام الذاكرة الافتراضية؛ يتوجب في كل مرة تنقل فيها البيانات أن تنسخ أيضاً في page cache لأنه يعالج بواسطة نظام الذاكرة الافتراضية وبهذا يمكن نقل البيانات إلى التطبيق الذي طلبها.

وعكس ذلك في الكتابة، حيث يتم تعدل البيانات في page cache ثم تنسخ إلى Buffer cache لتكتب فيما بعد على القرص.

إن أداء هذه العمليات بطيء كما أن عملية النقل للبيانات مرتين مكلفة من حيث:

1. استهلاك مساحة الذاكرة (ضعفي حجم بيانات الملف)؛ وبذلك تقل مساحة الذاكرة المتاحة للتطبيقات.
2. إضاعة وقت المعالج (فترة نقل البيانات بين الـــ page cache و Buffer cache)

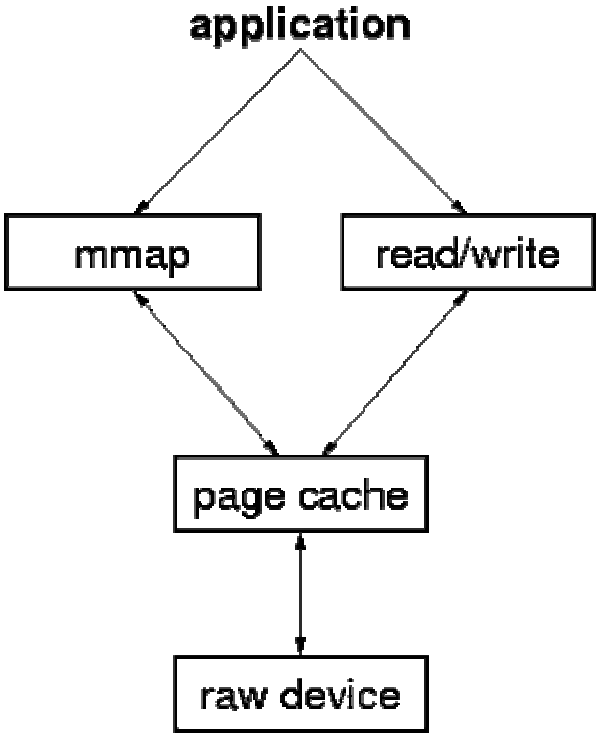
كما أن وجود نسختين من نفس البيانات قد يؤدي إلى حدوث عدم توافق (مثلاً: تطبيق يعدل على بيانات ملف في page cache وخلال ذلك يصدر تطبيق آخر أمر قراءة لنفس الملف ، سوف يقرأ من Buffer cache معلومات غير محدثة !)



ولحل هذه المشاكل تم التوصل إلى ابتكار نظام Unified Buffer Cache

4 . Unified Buffer Cache(UBC)

في هذا النظام يتم نقل بيانات الملف المطلوبة دائماً بغض النظر عن كيفية طلبها إلى page cache مباشرة بدون الحاجة إلى وجود buffer cache وبهذا يمكن لنظام الذاكرة الافتراضية إدارات بيانات الملف مباشرة .



المصادر:

- http://searchstorage.techtarget.com/sDefinition/0,,sid5_gci211963,00.html
- http://www.faqs.org/docs/linux_admin/buffer-cache.html
- http://www.usenix.org/publications/library/proceedings/usenix2000/freenix/full_papers/silvers/silvers_html/
- Operating system concepts-6th ed(page 434-435)