How can I tell whether a file is on an SSD?

devblogs.microsoft.com/oldnewthing/20201023-00

October 23, 2020

Raymond Chen

You might want your program to change its behavior depending on whether the file you are operating on is on an SSD or not. Maybe you'd use `PrefetchVirtualMemory` to get the contents of a memory-mapped file into memory more efficiently if the file is on a hard drive, but not bother if the file is on an SSD, since the SSD can produce the data quickly enough anyway.

```
bool IsFileOnSsd(PCWSTR filePath)
{
  wil::unique_hfile volume = GetVolumeHandleForFile(filePath);

  STORAGE_PROPERTY_QUERY query{};
  query.PropertyId = StorageDeviceSeekPenaltyProperty;
  query.QueryType = PropertyStandardQuery;
  DWORD bytesWritten;
  DEVICE_SEEK_PENALTY_DESCRIPTOR result{};

  if (DeviceIoControl(volume.get(), IOCTL_STORAGE_QUERY_PROPERTY,
      &query, sizeof(query),
      &result, sizeof(result),
      &bytesWritten, nullptr)) {
    return !result.IncursSeekPenalty;
  }
  return false;
}
```

This takes advantage of the trick we learned last time where you can make a storage query against a volume, and it will report the answer if the volume has a single extent.

We aren't so much checking whether it's on an SSD drive as we are checking whether seeks are free. That is true for SSDs, but it's also true for RAM drives. But RAM drives are even faster than SSDs, so I think it's okay to treat them as "super-awesome SSDs".

The `GetVolumeHandleForFile` function we wrote a few days ago will throw if the file is remote (on a network). We probably want to report network files as "not on an SSD", because even if they are on an SSD on the server, the network transmission cost will make it feel slow.

```
wil::unique_hfile GetVolumeHandleForFile(PCWSTR filePath)
{
  wchar_t volumePath[MAX_PATH];
  THROW_IF_WIN32_BOOL_FALSE(GetVolumePathName(filePath,
                              volumePath, ARRAYSIZE(volumePath)));

  wchar_t volumeName[MAX_PATH];
  if (!GetVolumeNameForVolumeMountPoint(volumePath,
                              volumeName, ARRAYSIZE(volumeName))) {
    return {};
  }

  auto length = wcslen(volumeName);
  if (length && volumeName[length - 1] == L'\\')
  {
    volumeName[length - 1] = L'\0';
  }

  wil::unique_hfile result{ CreateFile(volumeName, 0,
              FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
              nullptr, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, nullptr) };
  THROW_LAST_ERROR_IF(!result);
  return result;
}
```

We would then check whether a volume was gotten:

```
bool IsFileOnSsd(PCWSTR filePath)
{
  wil::unique_hfile volume = GetVolumeHandleForFile(filePath);
  if (!volume) return false;

  STORAGE_PROPERTY_QUERY query{};
  query.PropertyId = StorageDeviceSeekPenaltyProperty;
  query.QueryType = PropertyStandardQuery;
  DWORD bytesWritten;
  DEVICE_SEEK_PENALTY_DESCRIPTOR result{};

  if (DeviceIoControl(volume.get(), IOCTL_STORAGE_QUERY_PROPERTY,
      &query, sizeof(query),
      &result, sizeof(result),
      &bytesWritten, nullptr)) {
    return !result.IncursSeekPenalty;
  }
  return false;
}
```

As we noted last time, the query against a volume will fail if the volume spans multiple physical disks. If you have a volume that spans multiple SSDs, this function will nevertheless report that it isn’t an SSD.

So we probably would be better off checking the SSD-ness of every physical drive in the volume. That's a lot of work, so I'm going to cheat and check just the first physical drive in the volume, on the theory that when people create multi-drive volumes, they're going to be drives of similar performance characteristics.

```
bool IsFileOnSsd(PCWSTR filePath)
{
  wil::unique_hfile volume = GetVolumeHandleForFile(filePath);
  if (!volume) return false;

  wil::unique_hfile disk =
    GetFirstPhysicalDiskHandleForVolume(volume.get());
  if (!disk) return false;

  STORAGE_PROPERTY_QUERY query{};
  query.PropertyId = StorageDeviceSeekPenaltyProperty;
  query.QueryType = PropertyStandardQuery;
  DWORD bytesWritten;
  DEVICE_SEEK_PENALTY_DESCRIPTOR result{};

  if (DeviceIoControl(disk.get(), IOCTL_STORAGE_QUERY_PROPERTY,
      &query, sizeof(query),
      &result, sizeof(result),
      &bytesWritten, nullptr)) {
    return !result.IncursSeekPenalty;
  }
  return false;
}
```

Bonus chatter: Many people cheat even further and also assume that the volume is mounted as a drive letter. In that case, obtaining the volume handle for the file is a simple matter of opening `\\.\X:`, where `X:` is the drive letter of the file you are interested in.

Raymond Chen

Follow