I ran multiple simulations with varying number of steps set in the mdp-file using -multidir but all simulations ended at the same time, i.e. at the wrong step count for some runs.

It is unclear what the expected behaviour of a multi-simulation should be if the user supplies any of the possible non-uniform distributions of init_step and nsteps, sourced from any of .mdp, .cpt or command line. Instead, we report on the non-uniformity and proceed. It's always possible that the user knows what they are doing. In particular, now that multi-simulations are no longer explicitly coupled, any heterogeneity in the execution environment will lead to checkpoints and -maxh acting at different time steps, unless a user-selected algorithm requires that the simulations stay coordinated (e.g. REMD or ensemble restraints).

In the implementation of signalling, we have stopped checking gs for NULL as a proxy for whether we should be doing signalling at that communication phase. Replaced with a helper object in which explicit flags are set. Added unit tests of that functionality.

Improved documentation of check_nstglobalcomm. mdrun now reports the number of steps between intra-simulation communication to the log file.

Noted minor TODOs for future cleanup.

Added some trivial test cases for termination by maxh in normal-MD, multi-sim and REMD cases. Refactored multi-sim tests to make this possible without duplication. This is complicated by the way filenames get changed by mdrun -multi by the former par_fn, so cleaned up the way that is handled so it can work and be re-used better. Introduced mdrun integration-test object library to make that build system work a little better. Made some minor improvements to Doxygen setup for integration tests.

Fixes #860, #692, #1857, #1942.

Change-Id: I5f7b98f331db801b058ae2b196d79716b5912b09
Yes, that's a "feature." (Also present for -multi.) Some people think it is important to use the minimum number of resources regardless of what the user explicitly asked for. I disagree, but in practice, they have a point because there is no code that lets us continue running one simulation while the others either exit or wait, because we have to keep doing inter-simulation MPI communication to coordinate behaviour like Ctrl-C and -maxh.

Sigh... Another example where we try to outsmart the user by overriding what they explicitly ask for - this has to stop.

What happens if the simulations would finish within a minute of each other? Would we still kill all simulations except for the first one to finish?

I would suggest we try and reset this. If somebody is passionate about not using too many resources, they should use the -maxh option rather than demanding everybody else should suffer.

Some old discussion and code at https://gerrit.gromacs.org/#/c/4312/. I could look into resurrecting that, but it will need some work because various context in do_md() has changed. If so, my memory is that multi-simulations will now be decoupled, and now it becomes simple to implement "do what the user said."

Any feature to try to detect inefficiency in that policy in practice would be something for another time (probably never).

Related to Bug #692: Frequency of checking for inter-simulation signalling is too high for large-scale parallel REMD added

An impending update to https://gerrit.gromacs.org/#/c/5899/ will fix the issue Viveca reported, by keeping uncoupled simulations uncoupled, while permitting algorithms to require coupling.

https://gerrit.gromacs.org/#/c/5899/11 resolves this - an uncoupled multi-simulation does whatever number of steps is asked for in each simulation, noting any difference that exists. This fix has become feasible only because there is no "always on" coupling of multi-simulations for checkpointing or signal handling, so it does not matter to the other simulations if one of them terminates early.