BOTTLENECKS ADDRESSED AND CHANGES MADE

No.	Location	Bottleneck	Change Made
1	Nucleus.cpp(CNucleus::getSumTl())	storeSub.push_back() caused heavy reallocations and copies	Reused a thread_local buffer, reserved capacity using (lMax - lMin + 1), and replaced push_back with emplace_back
2	Nucleus.cpp(CNucleus::getSumTl())	Repeated calls to getTl(L, ek*scale, temp) for same inputs	Added a small local cache to store computed getTl values and reused them
3	Nucleus.cpp (decay-product generation)	Heavy new/delete churn for CNucleus objects during decay	Introduced a recycle pool (acquire() / release()) to reuse CNucleus objects instead of allocating each time
4	Nucleus.cpp (decay-product generation)	Deep recursive decay traversal causing function call overhead	Replaced recursive decay traversal with an explicit stack-based loop
5	Project-wide	Extensive use of pow() in performance-critical code paths	Replaced pow() with equivalent direct arithmetic (e.g., x*x)
6	Nucleus.cpp (constructors & initialization)	Duplicated constructor logic and repeated setup code	Centralized all setup logic into initializeDefaults() and initialize(...)
7	Nucleus.cpp (decay-product handling)	Frequent temporary vector allocations across events	Reused decay-product vectors across events instead of recreating them
8	gm_mwExecFusion.cpp,gm_mwExecCompound.cpp	Throttled progress + cancel handling	Updated progress bar only every 10 iterations to reduce UI overhead

TIMES

	old time(s)	new time(s)	Speed Incerase(%)
Compound Nucleus Decay			
Decay events.= 1000	7.07	4..30	39.18%
No. Of decay events.= 3000	21.51	12.61	41.37%
A=195,J=50, Events= 3000	23.63	15.43	34.68%
Average speed increase: 38.41% faster, 1.6× faster			
Fusion Reaction			
Default	10.23	5.67	44.57%
Events= 1000	20.5	11.38	44.49%
A=27,100 MeV	7.46	3.94	47.18%
A=27,200 MeV	11.28	7.06	37.41%
Average speed increase: 43.91% faster, 1.8× faster			