Why does ghOSt support a spinning global agent model?

For centralized scheduling, there is a single global agent polling a single message queue and making scheduling decisions for all CPUs. The global agent must continuously spin to ensure that the overhead of scheduling is low. Otherwise, it might be scheduled (e.g. queue needs to wake up one or more agents when messages arrive) to read the message queue, and the overhead will be high.

Why may this be a better solution than having ghOSt maintain a single global runqueue on which per-CPU agents enqueue/dequeue tasks?

1. This lacks global information. Since each CPU only has its own runqueue, the system cannot make scheduling decisions based on a wider perspective of the system.
2. The overhead is high. Because message delivery to a per-CPU agent consists of adding the message to the queue, context switching to local agent, and dequeuing the message, while message delivery to a global agent only consists of adding the message to the queue and dequeuing the message. The overhead of context switching is high, so the overhead of scheduling is high.

Some other thoughts. Don't count as answers to the question. I think this may also introduce extra work, but it depends on the probability of scheduling, it is hard to say whether this is beneficial or not. For the global runqueue approach, the system always needs to a) queue the task to the global runqueue, b) dequeue the task to the global runqueue, c) queue the task to the local runqueue and d) dequeue the task to the local runqueue. For the per-CPU approach, If scheduling is needed, the system needs to a) queue the task to the local runqueue, b) dequeue the task to the local runqueue, c) queue the task to the global runqueue, d) dequeue the task to the global runqueue, e) queue the task to the scheduled local runqueue and f) dequeue the task to the scheduled local runqueue. If scheduling is not needed, the system needs to a) queue the task to the local runqueue and b) dequeue the task to the local runqueue. So whether this approach introduces extra work depends on the probability of the system needing to schedule a task. If the probability is low, this approach may actually require less queue/dequeue.