I only have a sketch of a mining pool at the moment, and did not make decision on what is the best way to implement it, because it seems like non-trivial task. In the following description, I will touch on the 3 most important features of a mining pool and give an idea on how my pool would be unique.

PROFITABILITY

The main idea is to use a slightly modified implementation of Ethereum client to be able to produce as many different variation of a block on the current height as required. Each miner, connected to the mining pool receives a different variation of the block (i.e. having a distinct BlockHash).
This could be achieved by, for example, adding some extra dummy transaction with unique data payload.
The effect of such trick is that when multiple miners solve the block at the same time or near the same time, all of these blocks get relayed by the pool, and it collects one block reward and as also potentially as many uncle rewards as there were solved variations. Since all variations have different BlockHash, they will be counted as different blocks and qualify for reward.
I am not sure if any of the mining pools are already doing this, but I believe such pool can potently produce more uncles, and therefore, offer better profitability in the short term. In the long term, all the pool would try to adopt this trick.

DDOS RESILIENCE

DDOS resilience is achieved through a combination of several measures:
1. The pool requires that the miner's address specified in the URL, contained some minimum amount of Ether, for example 10.
2. Instead of giving each miner its own unique variation of the block, the pool would generate fixed number of variations and assigns them to the miners using some fast-to-compute hash function. Therefore, get_Work requests would always be served from a cache.
3. The pool keep a dictionary of certain number of last seen submitted PoW solutions, to fend off the attacks based on submitting the same solution many times. It disconnects any TCP/IP connection that tries to submit the same solution more than once.
4. The pool tries to find the optimal difficulty for the miner by submitting a very difficult piece of work at the beginning of the TCP/IP connection, then lowering it quickly but gradually until the appropriate difficulty level. This means that an attacker wanting to overrun the EtHash verification would have to spend considerable computing power at the beginning of each connection (because it would have to first partially invert a difficult SHA3 hash - the first line of defence). For the honest miners, there will be initial period where difficulty is too high, but then the situation should normalise quickly.

SECURITY OF THE PAYOUT FUND

Instead of keeping an internal database of owed payouts, and potentially replicating it across multiple locations, the pool keeps track of this information in a contract on the Ethereum blockchain. As it produces candidate blocks, it includes transactions that insert new records about submitted shares into the contract. In order to make it cheap, or even free, it accepts its own transaction at a lower or zero gas cost. I have not tested whether it would work.
The payouts are effected by the miners themselves, whenever they want to. The contract collects the mined rewards and has the records of the owed payouts, so the members can call the contract to claim their share any time. Payout contract can be designed in such a way, that even if the private key of the mining pool (the one which is used to sign off the records about the shares) is compromised, no payouts owed to that point in time are lost. Miners can monitor the contract for the signs of such hack (for example, if their profitability suddenly drops), and switch to another, uncompromised pool.